



Departamento de Informática

Universidad de Castilla-La Mancha

Tesis Doctoral

**MANTIS: Definición de un Entorno para la
Gestión del Mantenimiento de Software**

Doctorando: Francisco Ruiz González

Director: Dr. D. Mario G. Piattini Velthuis



Departamento de Informática

Universidad de Castilla-La Mancha

Tesis Doctoral

**MANTIS: Definición de un Entorno para la
Gestión del Mantenimiento de Software**

Ciudad Real (España), junio de 2003

Doctorando: Francisco Ruiz González

Director: Dr. D. Mario G. Piattini Velthuis

Dedicatoria:

A Prado, mi esposa, y a Irene y Violeta, mis dos hijas. Las tres personas más importantes en mi vida. Gracias por vuestra comprensión y apoyo; y por haberme infundido ánimos en los momentos de debilidad. Tengo claro que habéis sido las principales perjudicadas durante mucho tiempo por dedicarme a realizar esta tesis y espero compensaros en el futuro. Nunca olvidaré una pregunta de Irene, hace ya algunos años: Papá, y entonces ¿cuando vas a poder estar en casa con nosotras más tiempo?.

A mis padres, Pilar y Francisco, porque supisteis darme lo más valioso: vuestro ejemplo de tesón y lucha por salir adelante con ocho hijos, especialmente en los tiempos difíciles. Supisteis inculcarme el valor del estudio y, en gran parte, mis méritos académicos son los vuestros.

A mi numerosa colección de hermanos: Pilar, María José, Miguel Ángel, Antonio, Mari Ángeles, Mari Prado y Cristina.

A mis amigos. Espero veros más a menudo a partir de ahora, sobre todo los fines de semana.

Agradecimientos

A Mario Piattini, mi director de tesis, mi maestro y mi amigo. Ha sido un ejemplo constante y diario para mí. En estos años he tenido el privilegio de observar cómo ayuda, orienta, dirige, aconseja, decide y “dice que no”; siempre con delicadeza y profundo respeto a los demás. Muchísimas gracias por su comprensión.

A Isidro Ramos, que fue el primero en contagiarme el gusanillo de la investigación. Hace ya muchos años que tuve la fortuna de encontrármelo en el camino. El destino quiso que yo fuera el primer Director del Centro de Cálculo de la Universidad de Castilla-La Mancha, mientras él se dedicaba a levantarla de la nada desde la responsabilidad de Rector.

A Cristóbal Pérez, mi primer maestro y profesor de Informática en la universidad. Cuando años después volvimos a coincidir como colegas, supo darme auténticos consejos. He aquí dos joyas: “Seguro que tú puedes con ello”; “Es un honor para un profesor que un alumno le acabe superando”.

A mis compañeros del Grupo Alarcos por su colaboración y ánimos continuos. Su permanente preocupación por cómo me iba la tesis han sido para mí de una gran ayuda. Gracias a Macario, mi colega de mantenimiento (y de sufrimiento). Gracias a Félix, Aurora, Mar y Manuel Ángel por su importante aportación en algunos trabajos de la tesis. Gracias a Coral y Marcela por su ayuda para que la docencia no me imposibilitara el acelerón final necesario para acabar la tesis. Gracias a Eduardo por su ayuda para la composición final del documento.

A Eduardo, Jorge, Carlos, Juan Pablo, Pepe, Luis, Ester, Julián, Jesús y Manuel; mis colegas de la vieja generación de profesores de la antigua Escuela Universitaria. Supieron poner el hombro mientras nos tomábamos un café para darnos ánimos mutuos. ¡Lo importantes que han sido para mí esos pequeños ratos de charla!.

A Luis y Maria Dolores, ex-alumnos que colaboraron con su esfuerzo desarrollando algunos de los prototipos de la tesis.

A Prado, Irene y Violeta, ver sus caras al levantarme cada mañana me recarga las pilas.

*El mundo no está amenazado por las malas personas,
sino por aquellos que permiten la maldad.*

Albert Einstein

Índices.

Índice de Contenidos.

Índices.....	I
Índice de Contenidos.....	III
Índice de Figuras.....	XI
Índice de Tablas.....	XV
Resumen.....	XIX
Abstract.....	XXI
Códigos UNESCO.....	XXIII
Palabras Clave / Keywords.....	XXIII
1. Introducción.....	1
1.1. Justificación.....	3
1.1.1. Nuevos Paradigmas para el Mantenimiento.....	3
1.1.2. Enfoque en la Gestión del Mantenimiento.....	5
1.2. Propuesta: El Entorno MANTIS.....	6
1.3. Hipótesis y Objetivos.....	9
1.3.1. Objetivo Global.....	10
1.3.2. Objetivos Parciales.....	10
1.4. Marco de la Tesis.....	11
1.4.1. Organización Investigadora.....	11
1.4.2. Proyectos de I+D.....	11
1.4.2.1. MANTEMA.....	12
1.4.2.2. MPM.....	13
1.4.2.3. MANTICA.....	14
1.4.2.4. MANTIS.....	14
1.4.2.5. TAMANSI.....	15
1.4.2.6. MAS.....	15
1.4.3. Encuadre de las Materias Tratadas.....	15
1.4.3.1. En la Taxonomía de Computación de ACM.....	15
1.4.3.2. En el SWEBOK.....	18
1.5. Trayectoria Seguida.....	18
1.6. Organización de la Tesis.....	20
2. Método de Trabajo.....	23
2.1. Métodos de Investigación en Ingeniería del Software.....	25
2.1.1. Investigación Cualitativa: Investigación-Acción.....	25
2.2. Aplicación de Investigación-Acción en este Trabajo.....	31
2.2.1. Participantes.....	32
2.2.2. Proceso de Investigación vs Proyectos.....	32
2.3. Evaluación de las Herramientas y el Entorno.....	38
3. Tecnología de Proceso Software.....	41
3.1. Procesos y Tecnologías Software.....	43

3.1.1. Características de los Procesos Software.	43
3.1.2. Tecnología de Proceso Software.	45
3.1.3. Entornos de Ingeniería del Software.	46
3.1.3.1. Integración.	47
3.1.3.2. Orientación a Procesos.	50
3.1.3.3. Modelo de Servicios: ISO 15940.	52
3.1.3.3.1. Servicios de Gestión Técnica.	54
3.1.3.3.2. Servicios de Gestión del Proyecto.	55
3.1.3.3.3. Servicios de Gestión de Procesos.	57
3.1.3.3.4. Servicios de Soporte.	59
3.1.4. Modelos de Proceso Software.	59
3.1.4.1. Elementos Básicos.	62
3.1.4.2. Niveles y Dominios.	63
3.1.4.3. Vistas.	64
3.1.4.4. Sistemas de Procesos Software.	65
3.1.5. Evolución de Procesos Software.	67
3.1.6. Lenguajes de Modelado de Procesos.	68
3.1.6.1. Taxonomía.	71
3.1.6.2. Ejemplo de Lenguaje Formal: MVP-L.	73
3.1.6.3. Ejemplo de Lenguaje Semi-formal: UML.	75
3.1.7. El Rol Humano en los Procesos Software.	77
3.1.7.1. Interacción de Usuario.	78
3.1.7.2. Interacción Interpersonal.	79
3.2. Propuestas de Entornos.	80
3.2.1. PSEE.	81
3.2.2. ECMA.	87
3.2.2.1. Marcos de Trabajo para EIS.	87
3.2.2.1.1. Dimensiones de Servicios.	92
3.2.2.2. Soporte a Proyectos.	93
3.2.3. PCTE.	95
3.2.3.1. Arquitectura Estructural.	95
3.2.3.2. Arquitectura Funcional.	96
3.3. Modelos de Proceso para Mantenimiento.	99
3.3.1. Ciclo de Vida: ISO 12207.	101
3.3.1.1. Procesos Principales.	102
3.3.1.2. Procesos de Soporte.	103
3.3.1.3. Procesos Organizacionales.	104
3.3.1.4. Roles y Organizaciones participantes.	105
3.3.2. Modelo de Referencia para Procesos: ISO 15504.	107
3.3.3. El Proceso de Mantenimiento: ISO 12207 y 14764.	111
3.3.3.1. Actividades y Tareas.	112
4. Gestión del Mantenimiento	117
4.1. Introducción.	119
4.2. Estudio Bibliográfico.	120
4.2.1. Método Empleado.	120
4.2.1.1. Acotamiento de las Búsquedas.	121
4.2.2. Resultados y Conclusiones.	124
4.3. Propuestas.	127
4.3.1. Gestión Global del Proceso.	127
4.3.1.1. MÉTRICA 3.	127

4.3.1.2. Modelo de Kajko-Mattsson para Gestión de Problemas.	128
4.3.1.3. Modelo de Ciclo de Vida de Kung y Hsu.	129
4.3.1.4. Modelo de Mantenimiento de Stoecklin, Williams y Stoecklin.	130
4.3.1.5. Metodología MANTEMA.	131
4.3.1.6. Otras Propuestas.	131
4.3.2. Gestión del Proyecto.	132
4.3.2.1. Planificación.	133
4.3.2.2. Estimación del Esfuerzo de Mantenimiento.	134
4.3.2.2.1. Variantes de COCOMO.	134
4.3.2.2.2. Modelos Dinámicos.	135
4.3.2.2.3. Puntos-Función.	135
4.3.2.2.4. Gestión del Conocimiento.	136
4.3.2.3. Control de la Productividad.	136
4.3.3. Gestión de la Calidad.	137
4.3.3.1. Calidad de Servicio de Niessink y Van Vliet.	137
4.3.3.2. Otras Propuestas.	138
4.3.4. Gestión de Riesgos.	139
4.3.4.1. Criterios y Factores de Riesgo de Schneidewind.	140
4.3.4.2. Análisis de Riesgos de Sherer.	142
4.3.5. Mejora.	143
4.3.5.1. Perspectiva de Servicio de Niessink.	143
4.3.5.1.1. Mejora Basada en la Medida.	143
4.3.5.1.2. Mejora Basada en la Madurez.	144
4.3.5.1.3. Comentarios.	145
4.3.5.2. Modelo de Madurez para el Mantenimiento Correctivo.	145
4.3.5.3. Otras Propuestas.	146
4.3.6. Gestión de Recursos Humanos.	147
4.3.7. Infraestructura.	148
4.3.8. Medida.	149
4.3.8.1. Métricas de Estabilidad del Proceso.	149
4.3.8.2. Métricas de Mantenibilidad.	150
4.3.8.3. Otras Propuestas.	151
4.3.9. Reutilización.	153
4.3.10. Alineamiento Organizacional.	154
5. El Entorno MANTIS.	157
5.1. Introducción.	159
5.1.1. El Mantenimiento como Proceso de Negocio.	159
5.1.2. Un Entorno Extendido.	161
5.1.3. Características Generales de MANTIS.	162
5.2. Estructura General.	163
5.2.1. Clases de Herramientas.	163
5.2.1.1. Herramientas Conceptuales.	164
5.2.1.2. Herramientas Metodológicas.	165
5.2.1.3. Herramientas Técnicas (Software).	165
5.3. Arquitectura Conceptual.	166
5.4. Sistema de Procesos.	169
5.4.1. Procesos vs Proyectos.	172
5.5. Ontologías.	173
5.5.1. Uso en MANTIS.	173
5.5.2. Método de Diseño Ontológico.	175

5.5.2.1. Sistema de Representación de las Ontologías.	178
5.5.2.1.1. Ejemplo de Formalización: Concepto de Actividad.	179
5.5.3. Ontología del Mantenimiento.	181
5.5.3.1. Sub-ontología de los Productos.	183
5.5.3.2. Sub-ontología de las Actividades.	185
5.5.3.3. Sub-ontología de Organización del Proceso.	189
5.5.3.3.1. Procedimientos.	189
5.5.3.3.2. Gestión de Peticiones.	190
5.5.3.3.3. Problemas.	192
5.5.3.3.4. Integración de los Tres Aspectos.	193
5.5.3.4. Sub-ontología de los Agentes.	196
5.5.3.5. Aspectos Comunes.	198
5.5.4. Ontología de los Flujos de Trabajo.	199
5.5.4.1. Modelo Conceptual de los Flujos de Trabajo.	200
5.5.4.2. Estados de un Proyecto y sus Actividades.	202
5.5.4.3. Representación.	203
5.5.5. Ontología de la Medida.	206
5.5.5.1. El Proceso de Medición.	207
5.5.5.2. Descripción Conceptual de la Medición.	208
5.5.5.2.1. Ejemplo: Actualizar el Importe de un Servicio de Mantenimiento.	209
5.5.5.3. Representación.	211
5.6. Metamodelos.	216
5.6.1. Metamodelo Genérico de Proceso Software.	218
5.6.1.1. Paquete Básico.	221
5.6.1.2. Paquete de los Flujos de Trabajo.	223
5.6.1.3. Paquete Organizacional.	224
5.6.2. Metamodelo de la Medida.	224
6. Soporte a la Gestión del Mantenimiento	227
6.1. Introducción.	229
6.1.1. Gestión de la Complejidad del Mantenimiento.	230
6.2. Métodos para los Procesos de Gestión.	231
6.2.1. Gestión Global del Proceso.	231
6.2.1.1. Externalización de Servicios de Mantenimiento.	231
6.2.1.1.1. Externalización en MANTEMA 3.	232
6.2.1.1.1.1. Actividades y Tareas Iniciales Comunes.	233
6.2.1.1.1.2. Actividades y Tareas Finales Comunes.	235
6.2.1.1.1.3. Documentos para la Externalización.	235
6.2.1.1.1.4. Acuerdos de Nivel de Servicio.	238
6.2.1.2. Gestión del Conocimiento.	239
6.2.1.2.1. Sistema Basado en Agentes.	240
6.2.2. Gestión del Proyecto.	242
6.2.2.1. Planificación de Recursos.	243
6.2.2.1.1. Recursos Humanos para el Mantenimiento Correctivo Urgente.	243
6.2.2.1.1.1. Modelo Económico.	244
6.2.2.1.1.2. Ejemplo y Comentarios.	245
6.2.2.1.2. Predicción del Esfuerzo de Mantenimiento en Aplicaciones Antiguas.	246
6.2.2.1.2.1. Descripción del Experimento.	247
6.2.2.1.2.2. Principales Resultados.	247
6.2.2.1.2.3. Ecuaciones de Predicción.	249
6.2.3. Gestión de Riesgos.	250
6.2.3.1. Objetivos de Control para Proyectos de Mantenimiento.	250

6.2.3.1.1. Metodología CobiT para Auditoría de Sistemas de Información.....	250
6.2.3.1.2. Adaptación de CobiT a Proyectos de Mantenimiento.....	252
6.2.3.1.3. Propuesta de Objetivos de Control.....	253
6.2.3.2. Riesgos en la Externalización de Servicios de Mantenimiento.....	255
6.2.3.2.1. Identificación de los Riesgos.....	256
6.2.3.2.2. Priorización de los Riesgos.....	257
6.3. Métodos para los Procesos de la Organización.....	259
6.3.1. Mejora.....	259
6.3.1.1. Guía para Evaluar la Madurez de un Servicio de Mantenimiento.....	260
6.3.1.1.1. Preguntas del Cuestionario.....	260
6.3.1.1.1.1. Preguntas del Nivel 2.....	261
6.3.1.1.1.2. Preguntas del Nivel 3.....	264
6.3.1.1.2. Caso de Aplicación.....	266
6.3.1.2. Evaluación del Proceso de Mantenimiento.....	268
6.3.1.2.1. Resultados del Proceso de Evaluación.....	270
6.3.2. Recursos Humanos.....	271
6.3.2.1. Distribución de Recursos Humanos en una Cartera de Proyectos.....	271
6.3.2.1.1. Maximización del Beneficio.....	272
6.3.2.1.2. Restricciones.....	274
6.3.2.1.3. Método de Cálculo.....	274
6.3.3. Medida.....	275
7. Herramientas Software.....	277
7.1. Introducción.....	279
7.1.1. Arquitectura Software.....	279
7.1.1.1. Integración de las Herramientas Internas.....	281
7.1.1.2. Integración de Herramientas Externas.....	281
7.1.1.2.1. Uso de Sistemas de Gestión de Flujos de Trabajo.....	282
7.2. Herramientas Horizontales.....	286
7.2.1. Interfaz de Integración: MANTIS-Tool.....	286
7.2.2. Gestión del Repositorio: RepManager.....	286
7.2.2.1. Casos de Uso.....	289
7.2.2.2. Arquitectura de Tres Capas.....	292
7.2.2.3. Componente “Implementación MOF”.....	293
7.2.3. Gestión de la Base de Conocimiento: KM-MANTIS.....	294
7.2.3.1. Arquitectura del Sistema.....	294
7.2.3.1.1. Comunidades de Agentes.....	295
7.2.3.1.2. Colaboración entre Agentes.....	297
7.2.3.1.3. Responsabilidades de los Agentes.....	298
7.3. Herramientas Verticales.....	299
7.3.1. Gestión de Peticiones de Modificación: MANTOOL.....	299
7.3.1.1. Informes.....	301
7.3.2. Administración de Modelos y Meta-Modelos: METAMOD.....	302
7.3.2.1. Casos de Uso.....	304
7.3.2.2. Integración con otras Herramientas CASE.....	304
7.3.2.3. Arquitectura de Tres Capas.....	305
7.3.3. Métricas de Mantenibilidad: MANTICA.....	306
7.3.4. Estimación de Recursos: SREM.....	308
7.3.5. Gestión de Recursos: CREM.....	309
7.3.5.1. Casos de Uso.....	310
7.3.5.2. Arquitectura de Tres Capas.....	312
7.3.6. Evaluación de la Madurez: MAN-Quest.....	313

7.3.6.1. Casos de Uso.....	313
7.3.6.2. Interfaz de Usuario.....	314
8. Conclusiones.....	317
8.1. Análisis de la Consecución de Objetivos.....	319
8.1.1. Objetivos Parciales.....	319
8.1.2. Objetivo Global.....	322
8.2. Contraste de Resultados.....	323
8.2.1. Publicaciones Internacionales.....	324
8.2.2. Publicaciones Nacionales y Latinoamericanas.....	326
8.2.3. Otras Publicaciones sobre Mantenimiento.....	327
8.3. Líneas Abiertas de la Investigación.....	329
8.3.1. Mejora Continua del Mantenimiento.....	329
8.3.1.1. Basada en la Gestión de Conocimiento.....	330
8.3.1.2. Basada en la Medición.....	330
8.3.2. Mantenimiento Ágil del Software.....	331
8.3.3. Mejorar la Arquitectura y Tecnología del Entorno.....	331
Anexos.....	333
A. Publicaciones sobre Gestión del Mantenimiento.....	335
B. Metodología MANTEMA.....	339
B.1. Características Generales.....	339
B.1.1. Participantes.....	342
B.2. Estructurada Detallada de Actividades y Tareas.....	342
B.2.1. Actividades y Tareas Iniciales Comunes.....	342
B.2.2. Actividades y Tareas del Mantenimiento no Planificable.....	343
B.2.3. Actividades y Tareas del Mantenimiento Planificable.....	344
B.2.4. Actividades y Tareas Finales Comunes.....	344
B.2.5. Documentación.....	345
C. MOF: Meta-Object Facility.....	347
C.1. Arquitectura Conceptual.....	347
C.2. El Modelo MOF.....	349
C.2.1. Constructores para Metamodelado.....	350
C.2.1.1. Clases MOF.....	350
C.2.1.1.1. Atributos.....	350
C.2.1.1.2. Operaciones.....	351
C.2.1.1.3. Referencias.....	351
C.2.1.2. Asociaciones MOF.....	351
C.2.1.2.1. Finales de Asociación.....	352
C.2.1.2.2. Semántica de las Asociaciones.....	352
C.2.1.3. Paquetes MOF.....	353
C.2.1.3.1. Mecanismos para Modularización y Reutilización.....	353
C.2.1.4. Tipos de Datos MOF.....	354
C.2.1.5. Otros Constructores MOF.....	354
C.2.2. Relaciones entre los Diversos Constructores.....	356
D. XMI: XML Metadata Interchange.....	357
D.1. Relación entre XMI y MOF.....	358
D.2. Diseño de los DTD's XMI.....	359
D.2.1. Estructura General de un DTD en XMI.....	360
D.2.1.1. Encabezamiento: XMI.header.....	361
D.2.1.2. Contenido: XMI.content.....	362

D.2.1.2.1. Especificación de Clases.	362
D.2.1.2.2. Especificación de Atributos.	363
D.2.1.2.3. Especificación de Asociaciones.	363
D.2.1.3. Diferencias: XMI.differences.	363
D.2.1.4. Extensiones: XMI.extensions.	364
D.2.1.5. Referencias: XMI.reference.	364
D.3. Generación de los Documentos XMI.	364
D.3.1. Producción por Composición de Objetos.	365
D.3.1.1. Ejemplo de Aplicación: Grafos.	365
D.3.2. Producción por Extensión de Paquetes.	368
E. Formalismo REFSENO para Representación de Ontologías.	369
E.1. Primitivas Epistémicas.	369
E.1.1. Conceptos.	369
E.1.2. Atributos.	370
E.1.2.1. Atributos Terminales.	370
E.1.2.2. Atributos no Terminales.	371
E.1.3. Tipos.	373
E.1.4. Instancias.	374
E.1.5. Otras Primitivas.	374
E.2. Caso de Uso: Ontología de los Productos.	375
F. SPEM: Software Process Engineering Metamodel.	379
F.1. Características Generales.	379
F.2. Metamodelo.	381
F.2.1. Paquete Elementos Básicos.	381
F.2.2. Paquete Dependencias.	382
F.2.3. Paquete Estructura del Proceso.	383
F.2.4. Paquete Componentes del Proceso.	385
F.2.5. Paquete Ciclo de Vida del Proceso.	386
F.3. SPEM como Perfil UML.	387
G. Correspondencias entre los Niveles Conceptuales de MANTIS.	391
G.1. Correspondencias M3-M2.	391
G.2. Correspondencias M2-M1.	392
H. Manuales de Usuario.	395
H.1. Manual de RepManager.	395
H.1.1. Servicio GenerarDocumentoXMI.	395
H.1.2. Servicio CargarDocumentoXMI.	397
H.2. Manual de METAMOD.	397
H.2.1. Formularios.	398
H.2.2. Funcionalidad.	398
H.3. Manual de CREM.	402
H.3.1. Carteras.	403
H.3.2. Proyectos.	403
H.3.3. Peticiones.	404
H.3.4. Recursos.	404
H.3.5. Informes.	405
H.3.6. Asignaciones.	406
I. Ejemplos de Documentos DTD y XMI.	407
I.1. Documentos para Correspondencias M3-M2.	407
I.1.1. Documento XMI del Nivel M2.	407
I.1.2. Documento DTD del Nivel M2.	431
I.2. Documentos para Correspondencias M2-M1.	436

Apéndices.	441
1. Lista de Acrónimos.	443
2. Referencias Bibliográficas.	447

Índice de Figuras.

Figura 1-1. Esquema de la propuesta formulada en esta tesis.	6
Figura 1-2. Proyectos de I+D vs alcance de la tesis.	12
Figura 1-3. Partes del SWEBOK tratadas en esta tesis.	18
Figura 1-4. La trayectoria recorrida en relación con los niveles de capacidad de ISO 15504. ...	19
Figura 2-1 . Carácter cíclico de Investigación-Acción.	28
Figura 2-2. Dos dimensiones en Investigación-Acción en Sistemas de Información.	30
Figura 2-3. Participantes en la aplicación de Investigación-Acción.	31
Figura 2-4. Estructura multiciclo con bifurcación utilizada en MANTIS.	33
Figura 3-1. Proceso de producción vs proceso de gestión.	44
Figura 3-2. Impacto de la Tecnología de Proceso Software.	45
Figura 3-3. Propiedades de la Integración de Herramientas en un EIS.	49
Figura 3-4. Espacio tridimensional de la integración en un EIS.	50
Figura 3-5. Método para el modelado descriptivo de procesos.	61
Figura 3-6. Elementos básicos de un Modelo de Proceso.	63
Figura 3-7. Jerarquía funcional en un sistema de procesos software.	66
Figura 3-8. Diagrama de proceso en MVP-L.	74
Figura 3-9. Metodología para Modelado de PS basado en UML.	76
Figura 3-10. Dominios del Proceso Software según el marco conceptual de Downson.	77
Figura 3-11. Patrones de interacción interpersonal en el proceso software.	79
Figura 3-12. Arquitectura funcional de un PSEE.	82
Figura 3-13. Arquitectura de referencia para PSEE.	83
Figura 3-14. Relaciones entre Servicios, Herramientas y Tareas.	94
Figura 3-15. Grupos de servicios conceptuales y componentes software de un entorno real.	94
Figura 3-16. Clasificación de los estándares ISO para ingeniería del software.	100
Figura 3-17. Procesos del ciclo de vida del software según ISO 12207.	102
Figura 3-18. Roles y puntos de vista de los procesos del ciclo de vida.	106
Figura 3-19. Contexto de la estimación de procesos software.	107
Figura 3-20. Categorías y procesos en el modelo de referencia ISO 15504.	109
Figura 3-21. Peticiones de modificación y tipos de mantenimiento.	111
Figura 3-22. El proceso de mantenimiento según ISO.	112
Figura 4-1. Etapas en el ciclo de vida de mantenimiento propuesto por Kung y Hsu.	130
Figura 4-2. Evolución de los costes de mantenimiento en proyectos clásicos vs XP.	132
Figura 4-3. Un ejemplo de dependencia producto/proceso (PPD).	133
Figura 4-4. Modelo de "brechas" de calidad de servicio.	137
Figura 4-5. Modelo de proceso genérico para la mejora basada en la medida.	143
Figura 4-6. Visión sistémica del Mantenimiento: entidades e interrelaciones en un momento determinado.	150
Figura 4-7. Modelo de procesos de MwR (<i>Maintenance with Reuse</i>).	154

Figura 5-1. Aspectos del mantenimiento como proceso de negocio.	160
Figura 5-2. Estructura General del Entorno MANTIS.	164
Figura 5-3. Ejemplos de correspondencias entre niveles conceptuales en MANTIS.	168
Figura 5-4. Resumen del sistema de procesos de MANTIS.	170
Figura 5-5. Patrón " <i>Basic Process Structure</i> ".	171
Figura 5-6. Procesos versus Proyectos.	172
Figura 5-7. La ontología MANTIS es un filtro de conocimiento al definir modelos.	174
Figura 5-8. Organización de las ontologías de MANTIS.	176
Figura 5-9. Resumen de los factores de dominio que afectan al PMS.	181
Figura 5-10. Esquema básico de la ontología del mantenimiento.	182
Figura 5-11. Diagrama de la sub-ontología de los productos.	184
Figura 5-12. Diagrama de la sub-ontología de las actividades.	186
Figura 5-13. Sub-ontología de organización del proceso: los procedimientos.	190
Figura 5-14. Sub-ontología de organización del proceso: gestión de peticiones de mantenimiento.	191
Figura 5-15. Sub-ontología de organización del proceso: informes de investigación.	191
Figura 5-16. Diagrama global de la sub-ontología de organización del proceso.	192
Figura 5-17. Diagrama de la sub-ontología de los agentes.	196
Figura 5-18. Ejemplo de diagrama de flujo de trabajo.	201
Figura 5-19. Modelo de estados de las instancias de proyecto.	202
Figura 5-20. Estados y transiciones para instancias de actividad.	203
Figura 5-21. Diagrama de la ontología de los flujos de trabajo.	204
Figura 5-22. Modelo del Proceso de Medición propuesto en MANTIS.	208
Figura 5-23. Diagrama de la ontología de la medida.	213
Figura 5-24. Arquitectura general de metamodelado del Entorno MANTIS.	217
Figura 5-25. Arquitectura de metamodelos basada en dominios.	219
Figura 5-26. Relación entre definición y ejecución (reificación) de proceso.	220
Figura 5-27. Paquetes que forman el metamodelo de proceso software.	221
Figura 5-28. Metamodelo genérico de proceso software: paquete básico.	222
Figura 5-29. Metamodelo genérico de proceso software: paquete de flujos de trabajo.	223
Figura 5-30. Metamodelo genérico de proceso software: paquete organizacional.	224
Figura 5-31. Paquete del metamodelo de la medida.	225
Figura 6-1. Procesos organizacionales incluidos en MANTIS.	229
Figura 6-2. La externalización del mantenimiento en la metodología MANTEMA.	232
Figura 6-3. Fases de un proyecto en MANTIS.	243
Figura 6-4. Relación entre tamaño y esfuerzo de mantenimiento correctivo.	249
Figura 6-5. Las tres dimensiones del marco conceptual de CobiT.	251
Figura 6-6. Evaluación y mejora de procesos según ISO 15504.	267
Figura 6-7. Estructura del artefacto OUTPUT con los resultados de la evaluación.	270
Figura 6-8. El área de proceso clave de "Medición y Análisis" en CMMi.	275
Figura 7-1. Arquitectura software de MANTIS.	280
Figura 7-2. Las fases de diseño y de ejecución.	283
Figura 7-3. Modelos de interacción de MANTIS con un SGFT externo.	285

Figura 7-4. Esquema de funcionamiento de RepManager.	287
Figura 7-5. Ejemplos de los tres tipos de documentos XMI almacenados en RepManager.	288
Figura 7-6. Diagrama UML de casos de uso de RepManager.	290
Figura 7-7. Arquitectura de tres capas de RepManager.	292
Figura 7-8. Diagrama UML de casos de uso del sub-componente "Implementación MOF". ..	293
Figura 7-9. Arquitectura de agentes de KM-MANTIS.	295
Figura 7-10. MANTOOL: estado actual de una petición de modificación.	300
Figura 7-11. Ejemplo de formulario en MANTOOL: entrada de acciones correctivas.	301
Figura 7-12. MANTOOL: ejemplo de informe de tendencia.	302
Figura 7-13. METAMOD: Diagrama UML de casos de uso.	303
Figura 7-14. Exportación de metamodelos en formato MOF/XMI desde Rational Rose.	305
Figura 7-15. METAMOD: Aspecto del interfaz de usuario.	306
Figura 7-16. MANTICA: Formulario de comparación de modelos.	307
Figura 7-17. SREM: Pantalla principal.	308
Figura 7-18. CREM: Diagrama UML de casos de uso.	310
Figura 7-19. CREM: Diagrama de secuencia del caso de uso "añadir PM".	311
Figura 7-20. CREM: Diagrama de colaboración del caso de uso "calcular asignaciones".	311
Figura 7-21. CREM: arquitectura de tres capas.	312
Figura 7-22. MAN-Quest: Diagrama UML de casos de uso.	314
Figura 7-23. MAN-Quest: Formulario para crear cuestionarios.	315
Figura 7-24. MAN-Quest: Formulario para rellenar cuestionario.	315
 Figura B-1. Vista general de la estructura de actividades de MANTEMA.	 340
Figura B-2. Procesos de soporte con los que se establece un interfaz.	341
Figura B-3. Estructura del mantenimiento planificable.	345
 Figura C-1. Arquitectura de cuatro niveles para metadatos de MOF.	 348
Figura C-2. Jerarquía de herencia de las clases del Modelo MOF.	349
Figura C-3. Principales relaciones entre las clases del modelo MOF.	355
 Figura D-1. Intercambio abierto de metadatos entre distintos tipos de herramientas mediante XMI.	 357
Figura D-2. Relación entre MOF y XMI.	358
Figura D-3. Ejemplo de código XMI: una correspondencia M2-M1.	359
Figura D-4. Ejemplo de metamodelo para representación de grafos.	366
Figura D-5. Instancia del metamodelo y grafo que representa.	367
 Figura E-1. REFSENO: Ejemplo de representación alternativa de atributos no terminales.	 377
 Figura F-1. Modelo conceptual básico de SPEM.	 380
Figura F-2. Estructura de paquetes de SPEM.	380
Figura F-3. SPEM: paquete "Elementos Básicos".	381

Figura F-4. SPEM: paquete "Dependencias".	382
Figura F-5. SPEM: paquete "Estructura del Proceso".	383
Figura F-6. SPEM: paquete "Componentes del Proceso".	385
Figura F-7. SPEM: paquete "Ciclo de Vida del Proceso".	386
Figura F-8. Correspondencias simples entre el perfil SPEM y las clases base de UML.	388
 Figura G-1. Paquete básico adaptado del metamodelo de proceso software.	391
Figura G-2. Representación SPEM del mantenimiento no planificable de MANTEMA.	393
 Figura H-1. METAMOD: formulario para la descripción de nuevos metamodelos.	399
Figura H-2. METAMOD: formulario para editar propiedades de las clases.	399
Figura H-3. METAMOD: formulario de administración de metamodelos.	400
Figura H-4. METAMOD: Ver correspondencias entre elementos de diferentes niveles.	402
Figura H-5. CREM: Formulario para añadir peticiones de modificación.	403
Figura H-6. CREM: Formulario para añadir recursos para un intervalo de tiempo.	404
Figura H-7. CREM: Asignación de recursos a peticiones de modificación.	405
Figura H-8. CREM: Desglose de asignaciones de recursos por días y petición de modificación.	406

Índice de Tablas.

Tabla 1-1. Ficha resumen del proyecto MANTEMA.....	13
Tabla 1-2. Ficha resumen del proyecto MPM.....	13
Tabla 1-3. Ficha resumen del proyecto MANTICA.....	14
Tabla 1-4. Ficha resumen del proyecto MANTIS.....	14
Tabla 2-1. Investigación cualitativa vs cuantitativa.....	25
Tabla 2-2. Desarrollo del trabajo en los proyectos MANTEMA, MPM y MANTIS.....	37
Tabla 2-3. Diferentes técnicas de evaluación en el método DESMET.....	39
Tabla 3-1. Servicios de un Entorno de Ingeniería del Software.....	54
Tabla 3-2. Taxonomía de los principales sistemas de procesos software.....	66
Tabla 3-3. Clasificación de los servicios de usuario final en un Entorno de Soporte a Proyectos.....	95
Tabla 3-4. Documentos sobre los estándares ISO 12207 y 15504.....	100
Tabla 3-5. Actividades y tareas del proceso de mantenimiento según ISO.....	115
Tabla 4-1. Artículos y ponencias en publicaciones especializadas en mantenimiento.....	122
Tabla 4-2. Conferencias publicadas en la colección LNCS que han sido consultadas en detalle.....	123
Tabla 4-3. Otras conferencias cuyos programas finales fueron consultados.....	124
Tabla 4-4. Resumen de resultados del estudio bibliográfico.....	125
Tabla 4-5. Importancia asignada a la gestión del mantenimiento.....	126
Tabla 4-6. Actividades y tareas del mantenimiento de sistemas de información en MÉTRICA 3.....	128
Tabla 4-7. Factores de riesgo de los cambios en los requisitos que afectan a la mantenibilidad.....	141
Tabla 4-8. Niveles de madurez y áreas de proceso claves en ITS-CMM.....	144
Tabla 5-1. Niveles de la arquitectura conceptual basada en MOF.....	167
Tabla 5-2. Especificación de requisitos de la ontología de MANTIS.....	176
Tabla 5-3. Fuentes documentales utilizadas en la ontología del entorno MANTIS.....	177
Tabla 5-4. Esquema general de la ontología del mantenimiento: glosario de conceptos.....	183
Tabla 5-5. Esquema general de la ontología del mantenimiento: tabla de interrelaciones.....	183
Tabla 5-6. Sub-ontología de los productos: glosario de conceptos.....	184
Tabla 5-7. Sub-ontología de los productos: tabla de atributos.....	185
Tabla 5-8. Sub-ontología de los productos: tabla de interrelaciones.....	185
Tabla 5-9. Sub-ontología de las actividades: glosario de conceptos.....	188
Tabla 5-10. Sub-ontología de las actividades: tabla de atributos.....	188
Tabla 5-11. Sub-ontología de las actividades: tabla de interrelaciones.....	189
Tabla 5-12. Sub-ontología de organización del proceso: glosario de conceptos.....	194
Tabla 5-13. Sub-ontología de organización del proceso: tabla de atributos.....	195
Tabla 5-14. Sub-ontología de organización del proceso: tabla de interrelaciones.....	195
Tabla 5-15. Sub-ontología de organización del proceso: glosario de conceptos.....	197

Tabla 5-16. Sub-ontología de organización del proceso: tabla de atributos.....	198
Tabla 5-17. Sub-ontología de las actividades: tabla de interrelaciones.....	198
Tabla 5-18. Ontología del mantenimiento: tabla de clases de interrelaciones.	199
Tabla 5-19. Ontología de los flujos de trabajo: glosario de conceptos.	205
Tabla 5-20. Ontología de los flujos de trabajo: tabla de atributos.....	206
Tabla 5-21. Ontología de los flujos de trabajo: tabla de interrelaciones.	206
Tabla 5-22. Diferentes nomenclaturas para los conceptos de métrica y observación.	212
Tabla 5-23. Ontología de la medida: glosario de conceptos.	215
Tabla 5-24. Ontología de la medida: tabla de atributos.	215
Tabla 5-25. Ontología de la medida: tabla de interrelaciones.	216
Tabla 6-1. Actividades y tareas iniciales comunes en MANTEMA 3 (primera parte).	233
Tabla 6-2. Actividades y tareas iniciales comunes en MANTEMA 3 (segunda parte).....	233
Tabla 6-3. Adaptación de PMBOK al Sistema de Procesos de MANTIS.....	242
Tabla 6-4. Modelo económico diario.	244
Tabla 6-5. Resultado económico para el mantenedor con diferentes asignaciones a MUC.....	246
Tabla 6-6. Principales datos del caso de estudio.	247
Tabla 6-7. Principales estadísticas sobre el mantenimiento realizado.	247
Tabla 6-8. Predicciones para el mantenimiento correctivo urgente.	248
Tabla 6-9. Predicciones para el mantenimiento correctivo no urgente.	248
Tabla 6-10. Objetivos de Control de la Gestión de Cambios vs Actividades del PMS.....	253
Tabla 6-11. Cuestionario con los factores situacionales para externalización del mantenimiento.	257
Tabla 6-12. Cuantificación de los riesgos de la externalización a partir de los factores situacionales.	258
Tabla 6-13. Áreas de proceso clave para los niveles 2 y 3 de ITS-CMM.....	260
Tabla 6-14. Distribución de las PM's por tipos de mantenimiento.	266
Tabla 6-15. La evaluación y mejora del PMS en la arquitectura conceptual de MANTIS.	268
Tabla 6-16. Modelo para la evaluación y mejora: correspondencias M2-M1.....	269
Tabla 8-1. Breve estadística de las publicaciones de la tesis.	323
Tabla 8-2. Lista de publicaciones clasificadas por temas.	324
Tabla 8-3. Lista de publicaciones internacionales enmarcadas dentro del trabajo de la tesis... ..	326
Tabla 8-4. Lista de publicaciones nacionales y latinoamericanas enmarcadas dentro del trabajo de la tesis.	327
Tabla 8-5. Lista de publicaciones internacionales enmarcadas dentro del trabajo de la tesis... ..	329
Tabla A-1. Lista de publicaciones relacionadas con la Gestión del Proceso de Mantenimiento.	337
Tabla B-1. Estructura del mantenimiento no planificable.....	344
Tabla B-2. Lista de plantillas de documentos incluidas en MANTEMA.....	346
Tabla C-1. Mecanismos disponibles en MOF para la modularización y reutilización.....	354

Tabla D-1. Correspondencias M3-M2 en el ejemplo de metamodelado de grafos.	366
Tabla E-1. REFSENO: Ejemplo de una tabla de instancia.	374
Tabla E-2. REFSENO: Ejemplo de glosario de conceptos.	375
Tabla E-3. REFSENO: Ejemplo de una tabla de atributos de concepto: Artefacto.	375
Tabla E-4. REFSENO: Ejemplo de una tabla de atributos de concepto: Producto.	376
Tabla E-5. REFSENO: Ejemplo de una tabla de atributos de concepto: Versión.	376
Tabla E-6. REFSENO: Ejemplo de una tabla de tipos.	377
Tabla E-7. REFSENO: Ejemplo de una tabla de clases de atributos no terminales.	377
Tabla E-8. REFSENO: Extracto de un glosario de símbolos de ejemplo.	377
Tabla G-1. Ejemplos de correspondencias M3-M2.	392
Tabla G-2. Ejemplos de correspondencias M2-M1.	394

Resumen.

Dentro del ámbito de la Ingeniería del Software (y de la Informática en general), ha sido habitual prestar mucha más atención a los problemas relacionados con el desarrollo de nuevos productos software que a los relacionados con el mantenimiento posterior de dicho software. Esta falta de atención general hacia el mantenimiento ha cambiado ligeramente en los últimos años debido a eventos tan significativos como el efecto 2000 o la introducción del Euro, la nueva moneda de la Unión Europea. El cambio de tendencia no ha sido una moda pasajera (como ocurrió con los citados eventos), sino que se ha confirmado la necesidad de abordar los problemas especiales que surgen al hacer mantenimiento que no ocurren al hacer desarrollo, o que tienen unas características distintas que hacen que las soluciones no puedan ser las mismas. Esta nueva importancia asignada al mantenimiento no hace más que tener en cuenta mejor la realidad: todos los estudios confirman que el mantenimiento supone más del 60% (y en bastantes casos más del 80%) de los costes totales de un producto software a lo largo de su ciclo de vida. También es conocido que el coste medio de corregir un error durante la fase de mantenimiento es, aproximadamente, cien veces más alto que durante las etapas iniciales de requisitos.

En esta tesis se presenta una propuesta que pretende ayudar a mejorar la manera en que las organizaciones llevan a cabo el mantenimiento del software incidiendo en la mejora de la gestión de los proyectos. El punto de vista fundamental de esta propuesta es abordar el problema del mantenimiento desde una perspectiva de “proceso de negocio”, es decir, integrando los aspectos puramente ingenieriles (del software) con los organizacionales y de gestión. Se trata de avanzar hacia la consideración del mantenimiento del software como un aspecto clave para la competitividad de las organizaciones ya que, como han señalado diversos investigadores, éste es la parte central del cambio de los sistemas de información actuales.

La propuesta está basada en un nuevo concepto, definido ex profeso, llamado “Entorno de Ingeniería del Software (EIS) extendido”, que integra y amplía los conceptos tradicionales de metodología (básicamente, una colección de métodos relacionados) y de entorno de ingeniería del software (una colección de herramientas técnicamente integradas para automatizar los procesos de ingeniería del software). En este trabajo se ha considerado que un EIS extendido es una colección de herramientas conceptuales, metodológicas y técnicas cuya finalidad es poder abordar los procesos software (desarrollo o mantenimiento) desde una perspectiva global de proceso de negocio.

La propuesta desarrollada considera que la gestión de los proyectos software (en este caso, de mantenimiento) es un problema con múltiples caras. Por esta razón, la solución propuesta está basada en varias materias diferentes (según la clasificación de la Informática que hace la taxonomía de ACM): ingeniería del software (tecnología de proceso software: EIS, sistemas de procesos, herramientas CASE, ...), tecnologías y sistemas de información (arquitecturas conceptuales, metamodelado, ...), métodos de computación (ontologías, gestión del conocimiento, ...) y, por supuesto, gestión de proyectos.

En base a las ideas anteriores, en esta tesis se define el Entorno MANTIS, cuyas principales características son: integración por medio de herramientas, orientación a procesos, especialización en mantenimiento, y escalabilidad y adaptabilidad. Las tres clases de elementos del Entorno MANTIS son:

- 1) Un marco de trabajo conceptual para la gestión de proyectos de mantenimiento, que engloba una arquitectura conceptual multinivel (basada en el estándar OMG MOF), un

sistema de procesos basado en los estándares ISO, una colección de ontologías (del mantenimiento, de los flujos de trabajo, y de la medida) que pueden ser compartidas por las herramientas software y por los agentes humanos, y una colección de metamodelos representados en forma de documentos XML, DTD o XMI.

- 2) Una colección de procedimientos para los procesos de gestión y organizacionales (según son definidos en el modelo de ciclo de vida de ISO 12207), que complementan y mejoran la metodología MANTEMA, un completo y preciso modelo de cómo llevar a cabo el proceso de mantenimiento.
- 3) Una colección de prototipos definidos como componentes software del Entorno. La arquitectura software del sistema MANTIS establece tres tipos de componentes: verticales (herramientas para automatizar un determinado tipo de actividad), horizontales (que dan soporte a todo el Entorno), y externas (no incluidas directamente en el Entorno MANTIS, pero que pueden ser invocadas desde sus herramientas internas).

Los principales prototipos horizontales desarrollados han sido un gestor del repositorio integrado de datos y metadatos (en formatos XML y XMI), una herramienta para metamodelización, un gestor de la base de conocimientos, y un interfaz de integración. Los prototipos verticales incluyen, entre otros, un gestor de peticiones de modificación, una herramienta para la gestión de recursos humanos entre una cartera de proyectos de mantenimiento, o un gestor de cuestionarios de evaluación de la madurez de un servicio de mantenimiento. Por último, las principales herramientas externas consideradas en el Entorno MANTIS son los “Sistemas de Gestión de Flujos de Trabajo”, que se proponen como “motor de proceso” para la reificación de los procesos, es decir, para automatizar el seguimiento y control de los proyectos.

Abstract.

In Software Engineering (and also in Computer Science in general), it has been frequently to dedicate much more attention to the problems related with development of new products than to those related with its later maintenance. In the last years, this lack of general attention toward the maintenance has changed lightly due to events so significant as the Y2K or the Euro effects. The change of tendency has not been a fleeting fashion (like it happened with the mentioned events), but rather it has been confirmed the necessity to approach the specific problems that happen when maintenance is carry out and that don't happen when development is carry out, or that have different characteristics and, in consequence, the solutions cannot be the same ones. This new importance assigned to the maintenance is in order to consider better the reality: all studies confirm that the maintenance costs are more than two thirds (and in any cases more than 80%) of the total costs of a software product life cycle. It is also well-known that the cost of correcting an error during the maintenance stage is, approximately, one hundred times the cost that during the requirements initial stage.

In this thesis, a proposal is presented that seeks to help organizations to improve the way they carry out the software maintenance impacting in the project management improvement. The main point of view of this proposal is to approach the maintenance problem from a “business process” perspective, that is to say, integrating software engineering with organizational and management aspects. The idea is to advance toward the consideration of the software maintenance like a key factor for the competitiveness of the organizations because, like several researchers have pointed out, this work is the core part of the current information systems change.

The proposal is based on a new concept, named "extended Software Engineering Environment" (extended SEE), that integrates and extends the traditional concepts of methodology (shortly, a collection of related methods) and software engineering environment (a collection of technically integrated tools to automate the software engineering processes). In this work, it has been considered that an “extended EIS” is a collection of conceptual, methodological and technical (software) tools whose purpose is to be able to approach the software processes (development or maintenance) from a global perspective of business process.

The elaborated proposal considers that the software projects management (in this case, of maintenance) is a multidimensional problem with multiple faces. For this reason, the proposed solution is based on several different topics (according to the ACM taxonomy for the Computer Science): software engineering (software process technology: SEE, process system, CASE tools,...), information technologies and systems (conceptual architectures, meta-modeling,...), computing methodologies (ontologies, knowledge management,...) and, of course, project management.

Based on the previous ideas, the MANTIS Environment has been defined in this thesis with the following main characteristics: tool-based integration, process driven, maintenance specialization, and scalability and adaptability. The three classes of elements of the MANTIS Environment are:

- 1) A conceptual framework for the maintenance projects management that includes an multi-level conceptual architecture (based on the OMG MOF standard), a processes system based in the ISO standards, a collection of ontologies (of the maintenance, of the workflows, and of the measurement) that can be shared for the software tools and for

the human agents, and a meta-models set represented in form of XML, DTD or XMI documents.

- 2) A collection of procedures for the managing and organizational processes (as they are defined in the ISO 12207 life-cycle norm), that complement and improve the MANTEMA methodology, a complete and rigorous model in order to carry out the maintenance process.
- 3) A suite of prototypes which are defined as software components of the MANTIS Environment. The software architecture of the MANTIS system establishes three types of components: vertical (to automate a concrete activity type), horizontal (to automate a global service of the Environment), and external (not included into the MANTIS Environment, but which can be invoked by the internal tools).

The main horizontal developed prototypes have been the following: a data and metadata (in XML and XMI formats) repository manager, a meta-modeling tool, a knowledge base manager, and an interface of integration. The vertical prototypes include, among other, a modification requests manager, a tool for the human resources management among a maintenance projects portfolio, or a questionnaires manager for the maturity evaluation of a maintenance services. Lastly, the main external tools considered in the MANTIS Environment are the "Workflow Management Systems" which are proposed as "process engine" for the processes enacting, that is to say, to automate the projects monitoring and control.

Códigos UNESCO.

120317: Informática / Computer Science.

120318: Sistemas de Información, Diseño y Componentes / Information Systems, Design and Components.

Palabras Clave / Keywords.

Entorno de Ingeniería del Software.

Gestión de Proyectos Software.

Tecnología de Proceso Software.

Mantenimiento de Software.

Software Engineering Environment.

Software Project Management.

Software Process Technology.

Software Maintenance.

"La ignorancia afirma o niega rotundamente, la ciencia duda".
(Francois Marie Arouet - Voltaire)

1. Introducción.

Este primer capítulo introductorio busca, en primer lugar, *situar al lector en el qué, el porqué y el cómo* de esta tesis doctoral. En segundo lugar, pretende ser una ayuda inicial a la hora de decidir como abordar su lectura y comprensión. Para ello, se presentan los siguientes aspectos: importancia de la gestión en los proyectos de mantenimiento de software; opciones para mejorar dicha gestión; solución propuesta; hipótesis y objetivos del trabajo; encuadre de la tesis en cuanto a proyectos de I+D y en cuanto a las diversas disciplinas de conocimiento; evolución seguida durante su desarrollo; y, por último, estructura de este documento escrito.

1.1. Justificación.

Hace ya años que diversos autores vienen reclamando la necesidad de prestar más atención al mantenimiento del software (Pigoski, 1997), basándose en que todos los estudios demuestran que es la etapa del ciclo de vida de un producto software que más recursos consume (Piattini et al, 2000). Afortunadamente, la comunidad científica ha escuchado este llamamiento y, en los últimos años, han surgido iniciativas, reuniones, publicaciones, etc., dedicadas a abordar este asunto ¹. Glass (1998) es claro en este asunto cuando afirma que “el mantenimiento de software es una solución antes que un problema porque permite cambiar los artefactos antiguos añadiendo nueva funcionalidad”.

En el año 2000 se celebró en Limerick (Irlanda) la conferencia ICSE, una de las más prestigiosas a nivel internacional en su ámbito. En esta conferencia se incluyó una sesión dedicada al “futuro de la ingeniería del software” en la cual, Keith Bennett y Vaclav Rajlich, dos conocidos autores especializados en mantenimiento y evolución del software, presentaron un “roadmap” (Bennet y Rajlich, 2000) sobre mantenimiento de software que pretendía realizar una llamada de atención a la comunidad investigadora, no tanto para prestar más atención al tema, sino para abordarlo desde una perspectiva diferente. Estos dos autores identificaron los siguientes puntos de investigación clave, de cara al futuro:

- a) La producción de nuevas aproximaciones de gestión buscando mejorar la comprensión de las interrelaciones entre tecnología y negocio.
- b) ¿Cómo puede diseñarse el software para que sea más fácil de evolucionar después?.
- c) Más herramientas y métodos para la comprensión de los programas, tanto del código como de los datos.
- d) Una mejor formalización y conceptualización de la “mantenibilidad” (facilidad de mantenimiento); y ¿Cómo puede ser medida?.
- e) El desarrollo de un modelo de industria del software basada en servicios sustituyendo la perspectiva de producto.

Este tesis doctoral aborda aspectos de los puntos a) y e) especialmente, aunque también hace aportaciones, más limitadas, en los puntos c) y d). El autor de esta tesis lo resume diciendo que “*se trata de abordar el mantenimiento del software desde una perspectiva de proceso de negocio, más amplia que la meramente ingenieril utilizada hasta ahora*” (Ruiz y Piattini, 2001). En el capítulo 5 se precisa más esta idea.

1.1.1. Nuevos Paradigmas para el Mantenimiento.

Rajlich y Bennett (2000) también proponen un nuevo modelo de ciclo de vida del software cuyas etapas se establecen desde el punto de vista de las organizaciones y no desde el punto de vista de los ingenieros. Así, las etapas clásicas (requisitos, análisis, diseño, mantenimiento, etc.) se sustituyen por las siguientes:

¹ En el capítulo 4 se puede consultar un estudio bibliográfico sobre la gestión del mantenimiento para cuya realización se han utilizado las principales fuentes internacionales en el tema.

- *Desarrollo inicial:* Los ingenieros desarrollan la primera versión operativa del producto software.
- *Evolución:* Los ingenieros extienden las capacidades y funcionalidades del producto, probablemente de forma significativa.
- *Servicio:* Los ingenieros sólo reparan pequeños defectos y realizan cambios funcionales sencillos.
- *Finalización:* La organización decide no hacer más servicio del software aunque sigue buscando obtener ingresos de dicho producto tanto tiempo como sea posible.
- *Cierre:* La organización retira el producto del mercado y, si existe un producto sustitutivo, orienta a los usuarios a su uso.

Las cuatro últimas etapas ocurren después de haber entregado el producto a los clientes, es decir, se corresponden con el mantenimiento tal como es definido por los estándares ISO (ISO/IEC, 1995; 1998e). Esto está en consonancia con la realidad detectada por los estudios de mercado en el sentido de que el mantenimiento tiene una importancia capital para las empresas dedicadas al software. Además, cada una de las etapas anteriores difiere de las demás en algunos aspectos esenciales relacionados con las características del software y del equipo de trabajo: experiencia necesaria, arquitectura software, decaimiento del software (falta de coherencia arquitectural), y costes. Por tanto, cada etapa debería tener sus propias soluciones técnicas, procesos, necesidades de personal y actividades de gestión.

Algunos otros autores plantean un punto de vista similar pero para todos los procesos de ingeniería del software. Por ejemplo, Cockburn (2000) llama al resultado de aplicar esta idea una “*Big-M Methodology*”. Para esta autor, en el desarrollo o el mantenimiento de software se deben considerar, al menos, los siguientes factores: personas, roles, habilidades, equipos, herramientas, técnicas, procesos, actividades, hitos, entregables, estándares, medidas de calidad y valores de equipo.

Niessink (2000) destaca la dicotomía existente en el dominio de la ingeniería del software entre desarrollo y mantenimiento. Este autor afirma que las diferencias entre ambos son reales y que, por tanto, tienen consecuencias ciertas sobre los métodos, habilidades y herramientas necesarios en cada caso. Además, en línea con el último punto de investigación clave, de los cinco comentados al principio del capítulo, este autor propone abordar la mejora de los procesos de mantenimiento de software desde dos perspectivas de gestión diferentes: mejora basada en la medición, y mejora basada en madurez del servicio. En esta tesis se han tenido en cuenta ambas perspectivas y, por eso, se proponen métodos (ver capítulo 6) que las implementan.

Chapin (2002) recuerda que los sistemas de información automatizados son cambiados en su mayoría por medio de procesos de mantenimiento de software, aunque dichos sistemas de información engloban a más cosas que el software (también incluyen a personas, máquinas, materiales y métodos). Dice Chapin que “*esto resalta que la gestión del mantenimiento del software involucra a más cosas que el cambio del software; el mantenimiento hace cambios a un sistema de información y, por tanto, también al funcionamiento operativo de una organización*”.

1.1.2. Enfoque en la Gestión del Mantenimiento.

Las diversas propuestas para resolver o minimizar el problema del mantenimiento pueden dividirse en dos categorías (Piattini et al, 2000):

- a) Las soluciones de gestión u organizativas (en el capítulo 4 se realiza un detallado análisis del estado del arte en esta cuestión). Las postuladas en los últimos años han sido de los siguientes tipos:
 - Dotar de más y mejores recursos humanos para mantenimiento.
 - Gestionar la calidad buscando la mejora, no sólo de los productos, sino también de los procesos.
 - Mejorar la gestión de los procesos llevados a cabo durante el mantenimiento.
 - Mejorar la organización y gestión del equipo de trabajo.
 - Realizar más y mejor documentación de los cambios y hacer una mejor gestión de la configuración.
- b) Las soluciones técnicas al problema del mantenimiento del software son de dos clases: herramientas y métodos. Las primeras sirven para soportar de forma más efectiva y cómoda los segundos. Estas *herramientas* son diseñadas para ayudar al personal de mantenimiento a comprender los programas y a probar sus modificaciones para asegurar que no han sido introducidos errores. Muchas de estas herramientas son iguales o similares a las utilizadas para la prueba (test) del software: formateador, analizador estático, estructurador, documentador, depurador interactivo, generador de datos de prueba y comparador. Los principales *métodos* empleados en el mantenimiento del software son:
 - La *reingeniería*, que consiste en el examen y modificación de un sistema para reconstruirlo en una nueva forma (Bennett et al, 1990);
 - La *ingeniería inversa*, que es el proceso de analizar un sistema para identificar sus componentes y las interrelaciones que existen entre ellos, así como para crear representaciones del sistema en otra forma o en un nivel de abstracción más elevado (Chikofsky y Cross, 1990); y
 - La *reestructuración*, que consiste en la modificación del software para hacerlo más fácil de entender y cambiar o menos susceptible de incluir errores en cambios posteriores (Arnold, 1986). Se diferencia de la ingeniería inversa en que el software reestructurado tiene el mismo nivel de abstracción que el original.

En contra de lo que preconizaban los autores citados en el apartado 1.1.1, las soluciones técnicas han sido mucho más estudiadas y analizadas desde hace años que las soluciones de gestión. Esta situación ha significado que el esfuerzo en investigación ha ido en una dirección divergente con las necesidades reales de las organizaciones y empresas dedicadas a las tecnologías del software.

O'Neill (1997) realizó un estudio sobre las repercusiones del mantenimiento en la competitividad global de las organizaciones y llegó a una clara conclusión: “*La gestión eficiente del mantenimiento del software es una de las claves del éxito empresarial en la actualidad, ya que los sistemas de información y las tecnologías de la información son vitales para la salud de*

una organización. Las empresas que incrementen su habilidad para mejorar el mantenimiento de software como una competencia central mejorarán su posición competitiva”.

Por tanto, parece claro que, tal como propugnan los expertos señalados en el apartado 1.1.1, es necesario prestar más atención a los aspectos no ingenieriles del mantenimiento del software. El autor de esta tesis comparte plenamente esta opinión y, de hecho, **este trabajo intenta ser una aportación para la mejora de la gestión del mantenimiento**. Entre los tipos de soluciones antes resumidos, en este trabajo se incide en la mejora de la gestión de proyectos de mantenimiento de software, es decir, proyectos cuyo objetivo es prestar un servicio de mantenimiento de software. Ahora bien, este enfoque en los aspectos de gestión no debe significar una pérdida de rigor. Al contrario, se trata de aportar la rigurosidad propia de los métodos ingenieriles no sólo a los procesos puramente de ingeniería del software (desarrollo o mantenimiento) sino también a los otros procesos de soporte y organizacionales, que son completamente necesarios para mejorar la calidad y eficiencia en la realización de los proyectos software.

1.2. Propuesta: El Entorno MANTIS.

La propuesta presentada en esta tesis no implica sólo aspectos de gestión ya que también aboga por integrar, desde una perspectiva más amplia de proceso de negocio, la dimensión de gestión con la dimensión de ingeniería del software. Para ello, se ha definido un “entorno extendido para la gestión de proyectos de mantenimiento de software” que integra y amplía dos conceptos básicos y conocidos (ver Figura 1-1): metodología y “Entorno de Ingeniería del Software” (EIS).

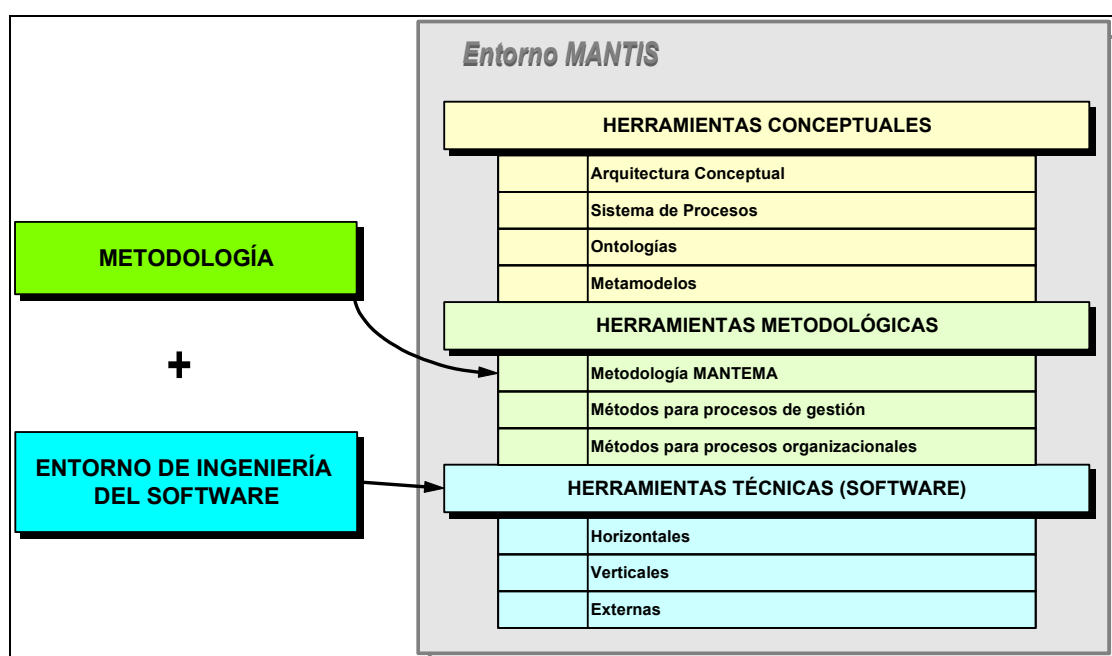


Figura 1-1. Esquema de la propuesta formulada en esta tesis.

En esta tesis se ha partido de la idea amplia de metodología tal y como la definieron Graham et al (1997): un conjunto de métodos o técnicas relacionados, junto con un modelo de procesos y una colección de entregables, métricas, herramientas y guías de gestión (incluyendo roles y organización del equipo de trabajo). En concreto, se ha partido de la metodología MANTEMA para el mantenimiento de software (Polo et al, 1999a) y se ha extendido para incluir diversos procedimientos (métodos, técnicas y guías) aplicables a los procesos de gestión y a los procesos organizacionales. Estos procesos son los así definidos en el modelo de ciclo de vida del software de la norma ISO 12207 (ISO/IEC, 1995). De esta manera se han definido las llamadas “*Herramientas Metodológicas*” (ver Figura 1-1) que se presentan en el capítulo 6.

Además, se ha considerado que es necesario dar soporte para la automatización de dichas herramientas metodológicas mediante una serie de herramientas software que deben funcionar de manera integrada. Para lograr dicha integración de forma eficiente se ha apostado por la idea de “Entorno de Ingeniería del Software” tal como es definido en la futura norma ISO 15940 (ISO/IEC, 2001b): una colección de herramientas que proporcionan un soporte automático, total o parcial, a las actividades de ingeniería del software. En concreto, se han desarrollado una serie de prototipos software útiles para automatizar diversas actividades propias del proceso de mantenimiento o de los procesos de gestión u organizacionales. Estos prototipos reciben el nombre de “*Herramientas Software*” (ver capítulo 7). Este EIS propuesto incluye también una arquitectura software que distingue entre herramientas especializadas en una actividad (verticales), las que dan soporte general (horizontales) y las externas pero utilizables desde el EIS. También se han definido unos requisitos tecnológicos que buscan la integración en varias dimensiones: datos, procesos, presentación, control y marco de trabajo. La principal base utilizada para la integración de las herramientas software es la propuesta PSEE (*Process-centered Software Engineering Environment*) de Derniame et al (1999b).

Por último, se ha definido un marco de trabajo conceptual que se ha considerado imprescindible para que las colecciones de herramientas metodológicas y técnicas formen de verdad un sistema de información automatizado ² integrado, útil para la gestión de proyectos de mantenimiento (ver Figura 1-1). Este marco de trabajo da el soporte conceptual a todas las herramientas metodológicas y técnicas y, por tanto, desempeña un papel central en la propuesta aquí presentada. En el capítulo 5 se detalla este marco de trabajo, justificando su utilidad y la razón de sus componentes, que son los siguientes:

- una arquitectura multinivel que facilita el manejo de la complejidad permitiendo representar información a diferentes niveles de abstracción (datos, modelos, metamodelos, etc.);
- un sistema de procesos claro y definido en base a los estándares ISO;
- una colección de ontologías (del mantenimiento, de los flujos de trabajo y de la medida) que permiten compartir el dominio del problema por todos los agentes que intervienen y por todas las herramientas;
- y una colección de metamodelos que permiten representar, en base a la arquitectura multinivel, los metadatos de las ontologías anteriores y el sistema de procesos y los datos de los proyectos de mantenimiento.

² En el sentido en que es definido en el informe FRISCO (IFIP, 1998): Un sistema de información automatizado es un sistema de información en el que las acciones son realizadas usando computadoras. Un sistema de información es una clase de sistema organizacional (personas que interactúan) que engloba la concepción de cómo los aspectos de comunicación y manejo de información son definidos y cómo funcionan, es decir, define explícita o implícitamente las acciones orientadas a la comunicación y al manejo de información.

En suma, en esta tesis se propone un Entorno extendido (y utilizamos la palabra entorno con E mayúscula para denotarlo) que engloba los tres tipos de herramientas citados. Así, **el Entorno MANTIS está formado por una colección de herramientas conceptuales, metodológicas y técnicas**, que son las anteriormente referidas (ver Figura 1-1) y que se presentan en detalle en los capítulos 5, 6 y 7 de esta tesis. Las principales características de este Entorno MANTIS son (ver capítulo 5 para más detalles):

- Integración por medio de herramientas;
- Orientación a procesos;
- Especialización en mantenimiento; y
- Escalabilidad y adaptabilidad.

Frente al diseño de entornos globales útiles para cualquier proceso software, la idea de diseñar un entorno específico para mantenimiento podría parecer poco práctica. Pero la especificidad del mantenimiento frente al desarrollo, ya comentada al principio de este capítulo, hace que los entornos existentes, diseñados todos ellos para el desarrollo, no sean directamente útiles para mantenimiento. Esto no implica que algunas de las herramientas diseñadas inicialmente para desarrollo no se puedan utilizar en mantenimiento. De hecho, una de las reglas básicas seguidas durante el desarrollo de esta trabajo ha sido tratar de evitar la duplicación de esfuerzos. Sólo se han desarrollado métodos o herramientas nuevos cuando los existentes no servían a las actividades y necesidades específicas del Entorno MANTIS.

La construcción de EIS de dominio específico puede tener importantes ventajas siempre y cuando no suponga violar la anterior regla de ahorro de trabajo. Además, al incorporar el marco de trabajo conceptual al Entorno MANTIS se posibilita un nivel de genericidad en las soluciones aportadas mucha mayor y, en consecuencia, una mayor reutilización y productividad en las herramientas (métodos o aplicaciones) incorporadas. Existen investigadores que ya han confirmado estas ventajas de los EIS de dominio específico basados en marcos de trabajo reutilizables. Por ejemplo, Lédeczi et al (2001), en su propuesta GME (*Generic Modeling Environment*), utilizan un marco de trabajo basado en una arquitectura conceptual, muy parecida a la de MANTIS, en la que los modelos y metamodelos son claves para la reutilización.

Tal como se comenta en el capítulo 5, en el Entorno MANTIS se ha considerado fundamental tener en cuenta que en los proyectos reales de mantenimiento participan personas y aplicaciones. Esto se concreta en dos aspectos:

- a) El conocimiento del dominio del problema debe poder ser compartido por agentes humanos y aplicaciones. Esta idea significa que el Entorno MANTIS incorpora una nueva dimensión de “*integración del conocimiento*”, en consonancia con las propuestas de Falbo et al (1998). Para ello son necesarias unas ontologías definidas de tal forma que puedan ser comprensibles a los participantes humanos, expertos o no en tecnologías de la información, y que puedan ser representadas y almacenadas en una computadora. Ambos requisitos se consiguen apostando por un sistema semi-formal basado en la propuesta REFSENO (Tautz y Von Wangenheim, 1998), adaptada para incluir diagramas UML de más fácil comprensión (capítulo 5).

- b) Se contempla la coexistencia de actividades manuales y automáticas o semiautomáticas. Esto se refleja en la sub-ontología de los agentes - presentada en el capítulo 5 - y en que la tecnología de “*flujos de trabajo*” desempeña un papel clave en el seguimiento y control de los proyectos, es decir, en la reificación de sus procesos (capítulo 7).

Por último, es interesante destacar que tres ideas centrales de la propuesta MANTIS son directamente extensibles a cualquier EIS, si se desea que sea algo más que una colección de herramientas software integradas a nivel tecnológico:

- Un EIS extendido es una ayuda para abordar proyectos software desde una perspectiva amplia de proceso de negocio, integrando los procesos de ingeniería del software (desarrollo y mantenimiento) y los procesos de gestión y organizacionales.
- Un EIS extendido está formado por tres tipos de herramientas: un marco de trabajo conceptual; una colección de metodologías, técnicas y guías; y una colección de herramientas software que automatizan las anteriores.
- El marco de trabajo conceptual debe incluir, al menos, una arquitectura multinivel; un sistema de procesos (necesario si el EIS es orientado a procesos, por ejemplo, si está basado en PSEE) y una colección de ontologías compartibles por personas y aplicaciones. Además, es deseable que también incluya un conjunto básico de metamodelos “centrales” en los cuales basarse para representar todos los modelos necesarios.

1.3. Hipótesis y Objetivos.

Esta tesis doctoral no es una tesis científica experimental en el sentido de que no busca verificar el cumplimiento en el mundo físico o natural que nos rodea de una hipótesis definida previamente. Se trata de una tesis de ingeniería (en concreto, de ingeniería informática) y, por tanto, la hipótesis formulada se refiere directamente a la posibilidad de diseñar, construir o mejorar un artefacto o sistema (físico o mental) que ayuda a las personas a resolver un problema o a minimizarlo. Tal como se comentó en apartados anteriores, el tipo de artefacto de esta tesis es una clase especial de sistema de información que ha sido llamado Entorno MANTIS. La hipótesis de estudio en esta tesis ha sido la siguiente:

Es posible definir un entorno extendido e integrado para ayudar a gestionar el mantenimiento de software desde una perspectiva amplia de proceso de negocio, que englobe la dimensión de ingeniería del software y la dimensión de gestión y organización.

1.3.1. Objetivo Global.

En consonancia con el título de la tesis, con la hipótesis anterior y también con la propuesta-solución resumida anteriormente, el objetivo global de este trabajo es el siguiente:

Definir un entorno extendido de ingeniería del software que permita abordar de forma integrada (desde una perspectiva amplia de proceso de negocio), la gestión de proyectos de mantenimiento del software.

Es interesante resaltar que la gestión se refiere a proyectos porque en el mundo real lo que se gestiona siempre son proyectos, es decir, “esfuerzos temporales emprendidos para crear un producto o servicio único” (PMI, 2000). En realidad, cuando se habla de gestión de procesos se refiere a la gestión de una parte del esfuerzo de un proyecto ya que, para obtener ese producto o servicio único, es mejor dividir el esfuerzo en varios sub-esfuerzos identificados como procesos. La propuesta PMBOK (*Project Management Body of Knowledge*), elaborada por el *Project Management Institute* (PMI) define un proyecto de manera muy genérica: “una serie de acciones que buscan lograr un resultado”. PMBOK establece dos categorías de procesos: los orientados al producto o servicio y los de gestión. En este trabajo los primeros están formados por el Proceso de Mantenimiento de Software (PMS) mientras que los segundos se subdividen en procesos puramente de gestión y procesos de organización (para más detalles, ver sistema de procesos de MANTIS en capítulo 5).

1.3.2. Objetivos Parciales.

Para definir mejor el alcance de este trabajo, el anterior objetivo global se ha desglosado en los siguientes objetivos parciales:

OP.1: Estudiar las tecnologías software, y en particular las llamadas “tecnologías de proceso software”, útiles para gestionar de forma rigurosa el mantenimiento de software.

OP.2: Estudiar las propuestas más actualizadas de métodos y técnicas para la gestión del mantenimiento de software.

OP.3: Ampliar la metodología MANTEMA con una colección de métodos, técnicas y guías útiles en la gestión de proyectos de mantenimiento de software.

OP.4: Desarrollar prototipos de herramientas software para los procedimientos del OP.3.

OP.5: Definir una arquitectura software adecuada para dar soporte automático integrado a los prototipos del OP.4.

OP.6: Definir un marco de trabajo conceptual que ayude a gestionar proyectos de mantenimiento de software de forma integrada.

En el capítulo 8 se incluye el análisis sobre la consecución de estos objetivos parciales (y del global) y los resultados o logros correspondientes.

1.4. Marco de la Tesis.

En este apartado se presenta el marco en el que se ha desarrollado la tesis. Primero se comenta la organización dentro de la cual el autor ha desarrollado este trabajo. Después se presentan los proyectos de I+D que han dado soporte económico parcial a este trabajo. Por último, se concreta el ámbito de conocimientos en el que se encuadra esta tesis.

1.4.1. Organización Investigadora.

El autor de esta tesis ha realizado el trabajo siendo miembro del **Grupo de I+D Alarcos** (<http://alarcos.inf-cr.uclm.es/>). Este grupo de investigación se creó en septiembre de 1997 por el profesor Mario Piattini y este autor. Rápidamente, el grupo fue creciendo tanto en personas como en proyectos. Desde sus comienzos, el mantenimiento del software fue una de sus líneas de trabajo. Así, este autor ha trabajado en el tema desde finales del año 1997, aunque en el alcance propio de la tesis los comienzos del trabajo fueron 2 años después, a finales de 1999. Prueba del rápido desarrollo del Grupo es que en la actualidad está formado por 13 profesores (6 funcionarios, 7 doctores y 10 con dedicación a tiempo completo). Todos ellos son profesores del Departamento de Informática de la Universidad de Castilla-La Mancha destinados en la Escuela Superior de Informática de Ciudad Real. Además, el grupo cuenta con otros 5 profesores colaboradores pertenecientes a universidades latinoamericanas, que están realizando sus tesis doctorales, y un número variable de becarios y estudiantes colaboradores.

1.4.2. Proyectos de I+D.

Los proyectos de I+D con financiación en alguna convocatoria oficial de ámbito regional o nacional, que han servido de soporte económico para el desarrollo del trabajo de esta tesis, son los siguientes:

- MANTEMA para definir una metodología para el mantenimiento (apartado 1.4.2.1);
- MPM, para la mejora del proceso de mantenimiento (apartado 1.4.2.2);
- MANTICA, para la medición de la mantenibilidad de esquemas de bases de datos (apartado 1.4.2.3);
- MANTIS, para definir el entorno extendido para la gestión integral del mantenimiento (apartado 1.4.2.4); y

Tal como se muestra en la Figura 1-2 el proyecto MANTEMA es previo al comienzo del trabajo propiamente dicho de esta tesis, pero supuso el inicio del esfuerzo en el problema del mantenimiento de software. Los tres proyectos siguientes están plenamente incluidos, por sus objetivos, dentro del alcance de la tesis, si bien MANTIS se puede considerar el más importante desde el punto de vista de los objetivos presentados en el apartado 1.3. Estos cuatro proyectos han tenido una serie de hitos a lo largo del tiempo que se pueden consultar en el capítulo 2,

junto con el método de trabajo seguido para integrar dichos proyectos de I+D con el trabajo propio de investigación de la tesis doctoral.

Además, existen otros dos proyectos que son continuaciones posteriores o paralelas en el campo del mantenimiento:

- TAMANSI, para definir nuevas técnicas avanzadas para el mantenimiento (apartado 1.4.2.5); y
- MAS, que busca definir métodos ágiles para el mantenimiento (apartado 1.4.2.6).

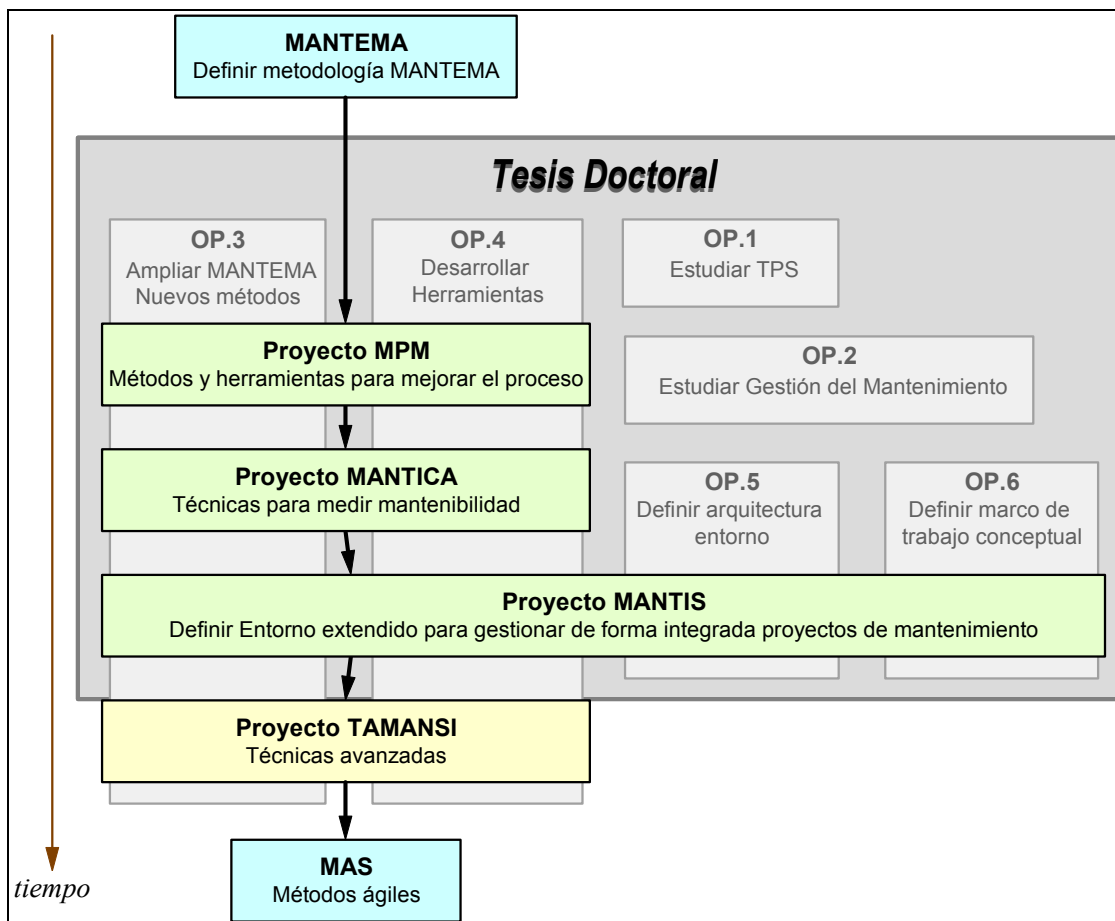


Figura 1-2. Proyectos de I+D vs alcance de la tesis.

A continuación se presenta un breve resumen de cada uno de estos proyectos.

1.4.2.1. MANTEMA.

Este proyecto fue el primero en el tema de mantenimiento, y también el primero comenzado por el Grupo Alarcos. El autor de esta tesis fue el investigador responsable. La idea del proyecto surgió por una petición de colaboración por parte de la empresa Atos ODS (que en ese momento se llamaba SLIGOS), una de las principales compañías de externalización de servicios software de Europa.

El objetivo de este proyecto fue “construir una metodología para el mantenimiento de software” definiendo las actividades y tareas del proceso de mantenimiento de una manera rigurosa. En esta metodología se contemplaron diversas tecnologías: lenguajes de tercera generación (COBOL), lenguajes de cuarta generación (NATURAL, FORMS, etc.), bases de datos relacionales (SQL) y lenguajes orientados a objetos (C++). En la primera fase se llevó a cabo un estudio en el que se analizaron los problemas asociados al mantenimiento de software y su repercusión en el ciclo de vida de los sistemas de información, así como las principales métricas y herramientas de mantenimiento. En la segunda fase se definió la metodología y se determinaron los casos de uso reales en las que sería probada. En la última fase se refinó la metodología con la experiencia de dichos casos de uso y se desarrolló una herramienta software para su soporte.

Título:	MANTEMA: Una Metodología para el Mantenimiento de Sistemas de Información
Entidad financiadora:	Atos ODS (artículo 11) y Ministerio de Industria y Energía (programa ATYCA T126-1997, T181-1998, T30-1999)
Importe de la ayuda (Euros):	48681.98 €
Entidades Participantes:	Universidad de Castilla-La Mancha y Atos ODS
Duración:	Julio/1997 a Julio/2000
Investigador Responsable:	Francisco Ruiz
Número de investigadores participantes:	5

Tabla 1-1. Ficha resumen del proyecto MANTEMA.

1.4.2.2. MPM.

El objetivo principal del proyecto MPM fue elaborar un conjunto de técnicas y métodos para mejorar el proceso de mantenimiento del software, obteniendo así sistemas de mayor calidad, especialmente por lo que respecta a su seguridad y mantenibilidad. Para ello, se definió un método de gestión de riesgos para el mantenimiento del software, se propusieron técnicas de mejora de procesos para el mantenimiento, y se elaboraron técnicas relativas a la gestión de los aspectos organizativos del mantenimiento. Además, se definieron un conjunto de métricas para controlar el proceso de mantenimiento y para la estimación de los recursos necesarios. Estas técnicas debían asegurar un mayor control sobre el proceso de mantenimiento, mejorando la productividad del personal de mantenimiento y disminuyendo los costes globales del proceso.

Título:	MPM: Mejora del Proceso de Mantenimiento
Entidad financiadora:	Atos ODS (artículo 11) y Ministerio de Industria y Energía (programa ATYCA FIT-070000-2000-307, TA15-1999)
Importe de la ayuda (Euros):	54091.09 €
Entidades Participantes:	Universidad de Castilla-La Mancha y Atos ODS
Duración:	Julio/1998 a Junio/2000
Investigador Responsable:	Mario Piattini
Número de investigadores participantes:	5

Tabla 1-2. Ficha resumen del proyecto MPM.

1.4.2.3. MANTICA.

Este proyecto fue el primero del Grupo Alarcos con financiación en una convocatoria oficial de ayudas a la investigación (los dos anteriores lo eran en convocatorias nacionales de ayudas a la innovación tecnológica). Este proyecto abordó un aspecto concreto dentro del mantenimiento de software en general, pero que se había descubierto que tenía una influencia considerable sobre el esfuerzo de mantenimiento en los sistemas de información. Se trataba de elaborar un conjunto de métricas que permitieran controlar la mantenibilidad (facilidad de mantenimiento) de esquemas de bases de datos objeto-relacionales (aunque posteriormente se extendió a otro tipo de bases de datos) a lo largo de todo el ciclo de vida de una base de datos y, especialmente, debido a su importancia, durante las fases de modelado conceptual y lógico. Para ello se analizaron las métricas software existentes y su aplicación a las bases de datos, así como el fundamento matemático para su definición (principalmente la teoría de la medida) y los métodos más adecuados para proceder a la validación de las métricas. Asimismo se definió un catálogo de métricas nuevas y se desarrolló una herramienta para su cálculo y visualización automáticos.

Título:	MANTICA: Definición de un Conjunto de Métricas para la Mantenibilidad de Bases de Datos Objeto-Relacionales
Entidad financiadora:	CICYT-FEDER (1FD97-0168 TIC)
Importe de la ayuda (Euros):	89088.02 €
Entidades Participantes:	Universidad de Castilla-La Mancha, Universidad del País Vasco, CIEMAT y Atos ODS
Duración:	Enero/1999 a Diciembre/2001
Investigador Responsable:	Mario Piattini
Número de investigadores participantes:	7

Tabla 1-3. Ficha resumen del proyecto MANTICA.

1.4.2.4. MANTIS.

Como se manifiesta por el acrónimo común, el proyecto MANTIS es el más directamente relacionado con esta tesis doctoral. El comienzo de este proyecto supuso el disponer de financiación específica para el objetivo general de la tesis. El alcance del proyecto MANTIS englobó de forma plena a los objetivos parciales OP.3, OP.4, OP.5 y OP.6 de esta tesis doctoral (ver Figura 1-2).

Título:	MANTIS: Entorno para el Mantenimiento Integral del Software
Entidad financiadora:	CICYT-FEDER (1FD1997-1608 TIC)
Importe de la ayuda (Euros):	57456.7 €
Entidades Participantes:	Universidad de Castilla-La Mancha, Universidad Rey Juan Carlos, Universidad Politécnica de Madrid, Atos ODS, Diputación Provincial de Ciudad Real
Duración:	Enero/2000 a Diciembre/2001
Investigador Responsable:	Mario Piattini
Número de investigadores participantes:	7

Tabla 1-4. Ficha resumen del proyecto MANTIS.

1.4.2.5. TAMANSI.

Este proyecto continua las propuestas del Entorno MANTIS con el fin de cubrir algunas necesidades detectadas que no quedaron resueltas en los proyectos anteriores. Arranca justo a continuación del alcance de esta tesis (ver Figura 1-2) ya que engloba la mejora basada en la medición y en la gestión del conocimiento, que son dos de las líneas de trabajo abiertas (ver capítulo 8). Pero también engloba aspectos que no se encuentran dentro del alcance de la tesis, principalmente, la necesidad de disponer de métodos para evaluar la mantenibilidad de aplicaciones web, y el desarrollo de nuevas técnicas (y sus respectivas herramientas) que permitan automatizar las pruebas de software avanzado (orientado a objetos, web, etc.) y simplificar así su mantenimiento.

Título:	TAMANSI: Técnicas Avanzadas para la Mantenibilidad de Sistemas de Información
Entidad financiadora:	Consejería de Ciencia y Tecnología cd Castilla-La Mancha (PBC-02-001)
Importe de la ayuda (Euros):	115900.00 €
Entidades Participantes:	Universidad de Castilla-La Mancha, Universidad de Sevilla, Universidad del País Vasco
Duración:	Junio/2002 a Diciembre/2004
Investigador Responsable:	Mario Piattini
Número de investigadores participantes:	19

1.4.2.6. MAS.

Este proyecto, titulado “Mantenimiento Ágil del Software”, ha sido presentado en la última convocatoria de proyectos de investigación científica y desarrollo tecnológico, del año 2003, del Ministerio de Ciencia y Tecnología. Todavía está pendiente de evaluación. Es un sub-proyecto del proyecto concertado “AgilWeb: Desarrollo y Mantenimiento Ágil de Aplicaciones basadas en Servicios Web”, común con la Universidad de Sevilla. El investigador responsable del sub-proyecto MAS es el profesor Macario Polo, miembro del Grupo Alarcos. Este proyecto pretende desarrollar una versión ágil de la metodología MANTEMA definiendo nuevas técnicas y herramientas en reingeniería, refactorización, pruebas, gestión del conocimiento y la experiencia, y gestión de riesgos.

1.4.3. Encuadre de las Materias Tratadas.

En este apartado situamos el alcance de esta tesis doctoral en relación con dos de las clasificaciones de conocimiento más conocidas en el campo de la Informática.

1.4.3.1. En la Taxonomía de Computación de ACM.

En primer lugar, y de forma muy general, se analizan las materias del “*extended Computing Classification System*” de ACM (2002a), que son abordadas en esta tesis. En la siguiente lista se indican en negrita las materias y submaterias tratadas en esta tesis. Las que son

parte importante de la tesis se destacan en color azul oscuro y subrayado. Se han excluido las subdivisiones de niveles 3 y 4 no tratadas en este trabajo.

- A. General Literature
- B. Hardware
- C. Computer Systems Organization
- D. Software/Software Engineering**
 - D.0 General
 - D.1 Programming Techniques
 - D.2 Software Engineering**
 - D.2.6. Programming Environments/Construction Tools**
 - D.2.6.c Integrated environments**
 - D.2.7 Distribution, Maintenance, and Enhancement**
 - D.2.7.b Corrections
 - D.2.7.d Enhancement
 - D.2.7.f Extensibility
 - D.2.7.g Maintainability
 - D.2.7.h Maintenance management**
 - D.2.7.i Maintenance measurement**
 - D.2.7.j Maintenance planning**
 - D.2.7.k Maintenance process**
 - D.2.8 Metrics/Measurement
 - D.2.8.c Process metrics
 - D.2.9 Management**
 - D.2.9.b Cost estimation
 - D.2.9.c Enactment**
 - D.2.9.e Organizational management and coordination
 - D.2.9.f Planning
 - D.2.9.k Project control & modeling**
 - D.2.9.m Risk management**
 - D.2.9.n Schedule and organizational issues
 - D.2.9.p Time estimation
 - D.2.12 Interoperability**
 - D.2.12.a Data mapping**
 - D.2.18 Software Engineering Process**
 - D.2.18.b Process infrastructure**
 - D.2.18.c Process measurement**
 - D.2.18.d Process definition**
 - D.2.18.e Software process models**
 - D.2.18.g Process implementation and change**
 - D.2.19 Software Quality/SQA
 - D.2.19.d Measurement applied to SQA and V&V
 - D.3 Programming Languages
 - D.4 Operating Systems
- E. Data
- F. Theory of Computation
- G. Mathematics of Computing
- H. Information Technology and Systems**
 - H.0 General
 - H.1 Models and Principles
 - H.2 Database Management**
 - H.2.4 Systems
 - H.2.4.p Workflow management
 - H.3 Information Storage and Retrieval**
 - H.3.3 Information Search and Retrieval**
 - H.3.3.d Metadata**
 - H.3.5 Online Information Services**
 - H.3.5.b Data sharing**
 - H.3.5.c DOM
 - H.3.5.e Web-based services
 - H.3.5.f XML/XSL/RDF**

- H.4 Information Technology and Systems Applications
 - H.4.1 Office Automation
 - H.4.1.g Workflow management
 - H.5 Information Interfaces and Representation (HCI)
 - H.m Miscellaneous
- I. Computing Methodologies**
 - I.0 General
 - I.1 Symbolic and algebraic manipulation
 - I.2 Artificial Intelligence**
 - I.2.4 Knowledge Representation Formalisms and Methods
 - I.2.4.a Agent communication languages
 - I.2.4.d Knowledge base management
 - I.2.4.i Representation languages
 - I.2.11 Distributed Artificial Intelligence
 - I.2.11.b Intelligent agents
 - I.2.11.d Multiagent systems
 - I.2.12 Intelligent Web Services and Semantic Web**
 - I.2.12.c Ontology design**
 - I.2.12.d Ontology languages**
 - I.2.13 Knowledge Management
 - I.2.13.a Knowledge acquisition
 - I.2.13.d Knowledge maintenance
 - I.2.13.e Knowledge modeling
 - I.2.13.h Knowledge retrieval
 - I.2.13.i Knowledge reuse
 - I.3 Computer Graphics
 - I.4 Image Processing and Computer Vision
 - I.5 Pattern Recognition
 - I.6 Simulation, Modeling, and Visualization
 - I.7 Document and Text Processing
 - I.m Miscellaneous
- J. Computer Applications
- K. Computing Milieux**
 - K.0 General
 - K.1 The Computer Industry
 - K.2 History of Computing
 - K.3 Computers and Education
 - K.4 Computers and Society
 - K.5 Legal Aspects of Computing
 - K.6 Management of Computing and Information Systems**
 - K.6.1 Project and People Management**
 - K.6.1.b Management techniques**
 - K.6.3 Software Management**
 - K.6.3.b Software maintenance**
 - K.6.3.c Software process**
 - K.7 The Computing Profession
 - K.8 Personal Computing
 - K.m Miscellaneous

La lista anterior permite confirmar algo que ya es evidente al definir los objetivos de la tesis. Este trabajo, dentro de la Informática, es de carácter multidisciplinar. Aunque está enfocado en el campo de la Ingeniería del Software, aborda un problema multidisciplinar y, por eso, la solución también lo es. Prueba de ello es que trata, con mayor o menor intensidad, materias o sub-materias englobadas en 4 de las 11 divisiones de primer nivel: D (Software/Ingeniería del software), H (Tecnologías y sistemas de información), I (Metodologías para computación), y K (Entorno social de la Informática).

1.4.3.2. En el SWEBOK.

De forma similar al apartado anterior, a continuación se analizan las materias abordadas en la tesis en relación con el cuerpo de conocimiento de la Ingeniería del Software definido por la propuesta SWEBOK (2001). En la Figura 1-3 se muestran en **negrita sobre fondo verde** las áreas y sub-áreas de conocimiento definidas en SWEBOK que son tratadas o utilizadas de forma significativa en este trabajo. Otra vez se comprueba, ahora dentro del ámbito más restringido de la ingeniería del software, que esta tesis aborda un problema complejo para el cual ha sido necesario estudiar, utilizar e integrar ideas y propuestas bastante diversas.

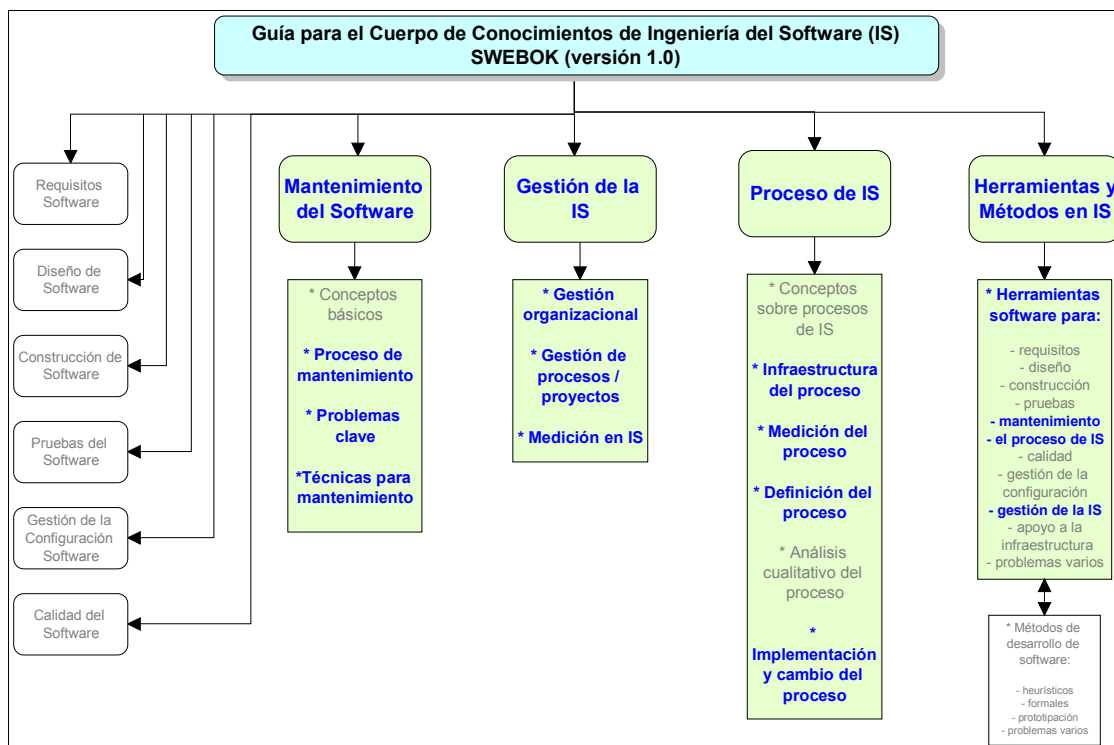


Figura 1-3. Partes del SWEBOK tratadas en esta tesis.

Como muestra la figura anterior, las cuatro áreas principales de la ingeniería del software contempladas en esta tesis son las siguientes: el “Mantenimiento del software” y la “Gestión de la I.S.” porque el dominio del problema es precisamente la gestión del mantenimiento; el “Proceso de I.S.” porque la solución propuesta está centrada, en gran parte, en la idea de proceso software; y las “Herramientas en I.S.” porque dicha solución propone un EIS extendido integrando diversas herramientas.

1.5. Trayectoria Seguida.

La evolución del trabajo a lo largo de los años que ha durado el desarrollo de la tesis está comentada en los proyectos de I+D, ya presentados, y se amplía en el capítulo 2.

La percepción que el autor ha tenido de en qué consistía su tesis doctoral también ha evolucionado a lo largo del tiempo. Las características del problema abordado, el método de trabajo seguido (capítulo 2) y, sobre todo, el propio proceso de aprendizaje por descubrimiento han hecho que los planteamientos y puntos de vista sobre este trabajo no hayan permanecido constantes durante los años que ha durado su desarrollo. Desde la idea original a finales del año 1999 de ampliar la metodología MANTEMA incorporando una colección de métodos adicionales, y las herramientas correspondientes, para algunos de los procesos de gestión hasta la propuesta de Entorno MANTIS, resumida anteriormente y presentada en detalle en el capítulo 5, se ha producido una evolución significativa. Sería largo y tedioso explicar cada paso de dicha evolución, pero podemos resumirla diciendo que conforme se ha ido estudiando cada vez más profundamente el problema de la gestión del mantenimiento, se ha ido descubriendo (en realidad confirmando) que es complejo y multidisciplinar. En un determinado momento se llegó a la conclusión de que el problema se debía tratar no sólo como el desarrollo de nuevos métodos y/o herramientas software, sino también de forma holística abordando su complejidad desde una perspectiva global. De ahí se derivó la necesidad de extender la ideas de metodología y de EIS y de integrarlas.

La evolución en la percepción del problema supuso, no sólo un dominio del problema cada vez más amplio, sino también la evolución en los puntos donde se focalizó el interés y, en consecuencia, el cambio en los objetivos de la tesis y de los nuevos proyectos de I+D. Es decir, también se fue ampliando el dominio de la solución.

Estas evoluciones se resumen en que se ha buscado la mejora del mantenimiento del software, incidiendo en la mejora de su gestión, siguiendo una evolución en línea con los modelos de mejora de procesos conocidos. Así, la metodología MANTEMA es una ayuda para que las organizaciones que realizan proyectos de mantenimiento de software puedan alcanzar el nivel 2 de madurez (gestionado) del modelo de capacidad definido en la norma ISO 15504 (ISO/IEC, 1998c). El Entorno MANTIS amplía los dominios del problema y de la solución, para ayudar a cumplir los requerimientos de los niveles 3 (establecido) y 4 (previsible). Además, propone líneas de trabajo nuevas para avanzar hacia el nivel 5 (optimizado). Por último, los nuevos proyectos inciden en dichas líneas para llegar al nivel 5 y también en nuevos métodos y herramientas para los niveles 3 y 4. La Figura 1-4 representa esta trayectoria de forma visual.

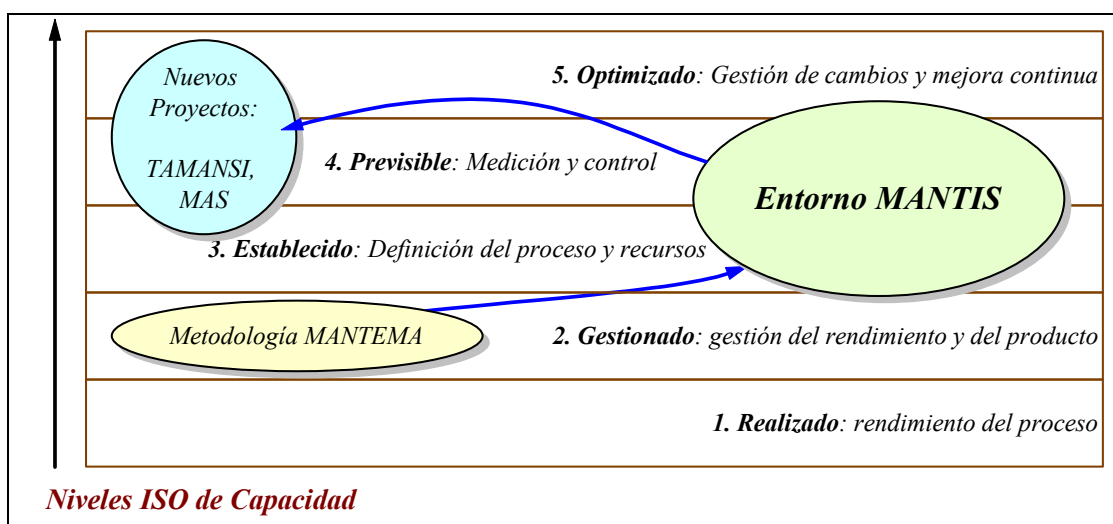


Figura 1-4. La trayectoria recorrida en relación con los niveles de capacidad de ISO 15504.

1.6. Organización de la Tesis.

Este documento está estructurado en ocho capítulos, nueve anexos y dos apéndices. Los contenidos de cada parte son los siguientes:

Capítulo 1 – Introducción: Este mismo capítulo.

Capítulo 2 – Método de Trabajo: Describe los métodos de trabajo y de investigación utilizados durante el desarrollo de la tesis: investigación-acción como método cualitativo de investigación en sistemas de información, y DESMET para evaluar entornos y herramientas software.

Capítulo 3 – Tecnología de Proceso Software: Presenta el estado del arte de las tecnologías útiles para mejorar los procesos software. Estas tecnologías se conocen con el nombre genérico utilizado como título del capítulo. En especial, se presentan los entornos de ingeniería del software, los modelos y lenguajes para el modelado de procesos, y los modelos y estándares útiles para definir y representar el proceso de mantenimiento.

Capítulo 4 – Gestión del Mantenimiento: Esta parte presenta el estado del arte de la gestión del mantenimiento de software. En primer lugar, incluye un estudio bibliográfico de las publicaciones relacionadas con el tema durante cuatro años y los resultados del análisis de más de 700 referencias. En segundo lugar, incluye una relación y análisis de las principales propuestas para gestionar el mantenimiento publicadas en los últimos años, clasificadas según el modelo de procesos del ciclo de vida del software de ISO.

Capítulo 5 – El Entorno MANTIS: Este capítulo describe en detalle la propuesta principal de la tesis. Una primera parte general se dedica a presentar las características y estructura del Entorno para, a continuación, presentar el marco de trabajo conceptual. En esta segunda parte se presentan la arquitectura conceptual multinivel, el sistema de procesos, la colección de ontologías propuestas (de mantenimiento, de los flujos de trabajo y de la medida) y, por último, el catálogo de metamodelos incluido (genérico de proceso software y de la medida).

Capítulo 6 – Soporte a la Gestión del Mantenimiento: Detalla las herramientas metodológicas de MANTIS, es decir, la colección de métodos, técnicas y guías definidas expresamente, dentro del alcance del Entorno, para llevar a cabo actividades de los procesos de gestión y de organización.

Capítulo 7 – Herramientas Software: Primero, se presenta la arquitectura software propuesta en el entorno MANTIS, indicando los tipos de herramientas internas incluidas y la integración de las externas (en especial, de los Sistemas de Gestión de Flujos de Trabajo). A continuación, se describen todas las herramientas internas clasificadas en las dos categorías arquitecturales: horizontales y verticales.

Capítulo 8 – Conclusiones: En este último capítulo se realiza un balance final. Incluye un análisis de la consecución de objetivos y las principales aportaciones del trabajo, la relación de publicaciones científicas conseguidas, y las líneas de trabajo que se dejan abiertas.

Anexos: Los nueve anexos incluidos amplían y precisan información o ejemplos útiles para la mejor comprensión de algunos de los aspectos presentados en los capítulos anteriores. La lista de anexos es la siguiente:

- o A - Publicaciones sobre Gestión del Mantenimiento.
- o B - Metodología MANTEMA.
- o C - MOF: Meta-Object Facility.
- o D - XMI: XML Metadata Interchange.
- o E - Formalismo REFSENO para Representación de Ontologías.
- o F - SPEM: Software Process Engineering Metamodel.
- o G - Correspondencias entre los niveles conceptuales de MANTIS.
- o H - Manuales de Usuario.
- o I - Ejemplos de documentos DTD y XMI.

Apéndices: Por último, los dos apéndices incluyen la lista de acrónimos utilizados en el texto y la lista de referencias bibliográficas citadas.

Investigar es como buscar el interruptor de la luz a oscuras, hay que ir probando, sin saber cuando se va a encontrar. Pero cuando uno lo encuentra y enciende la luz, todo el mundo ve sin problemas donde estaba el interruptor. (anónimo).

2. Método de Trabajo

En este capítulo presentamos un resumen de Investigación-Acción (*Action-Research*), el método de investigación cualitativo utilizado en este trabajo ya que los métodos empíricos son poco útiles para problemas de la naturaleza de los que esta tesis aborda de manera central.

También presentamos las características especiales que deben ser tenidas en cuenta al utilizar Investigación-Acción para hacer I+D en Sistemas de Información y, en especial, en Ingeniería del Software. Además, comentamos cómo hemos aplicado Investigación-Acción en este trabajo, explicando en particular los participantes que han intervenido y las etapas que hemos seguido durante su utilización con los diversos proyectos de I+D que han formado parte del marco de trabajo de esta tesis (ver capítulo 1).

Por último, explicamos la metodología que hemos utilizado para llevar a cabo una evaluación de las diversas herramientas construidas y del entorno en general.

2.1. Métodos de Investigación en Ingeniería del Software.

Comprender una disciplina implica aprendizaje, es decir, observación, reflexión y encapsulación de conocimiento, construcción de modelos (del dominio de aplicación, de los procesos para resolver problemas, etc.), experimentación, y evolución de los modelos con el tiempo (Basili, 2000). Este paradigma ha sido utilizado en campos tan diversos como la medicina, la física, o la industria manufacturera; pero las únicas diferencias entre unos campos y otros estriban en cómo se construyen y analizan los modelos, y cómo se lleva a cabo la experimentación.

Desde un punto de vista científico, las áreas de Sistemas de Información e Ingeniería del Software son de carácter experimental. La investigación en ellas se centra en conocer la naturaleza de los procesos, productos e interrelaciones entre ellos en el contexto de un sistema software o de un sistema organizacional (en el caso de Sistemas de Información). Gran parte de la investigación llevada a cabo en ambas áreas es de tipo cuantitativo y está basada en técnicas estadísticas. Sin embargo, en los últimos años los métodos de investigación cualitativos, y en especial Investigación-Acción, han merecido la atención y aceptación de la comunidad investigadora en Sistemas de Información (Avison et al, 1999), (Seaman, 1999); aunque ya hace bastantes años que se publicó la primera propuesta al respecto (Wood-Harper, 1985). La investigación cualitativa difiere de la cuantitativa es varios aspectos que se muestran resumidos en la Tabla 2-1.

Aspecto	Investigación Cualitativa	Investigación Cuantitativa
Propósito	proveer conocimiento sobre una organización y/o un problema y sus soluciones	generalizar los resultados desde un caso de estudio a la población total de interés
Orientación	a la verificación	al descubrimiento
Tamaño de la muestra	pequeño	grande
Recogida de datos	datos poco estructurados obtenidos mediante entrevistas, observaciones y discusiones en grupo	datos y técnicas muy estructurados
Análisis de datos	no estadístico	estadístico

Tabla 2-1. Investigación cualitativa vs cuantitativa.

2.1.1. Investigación Cualitativa: Investigación-Acción.

Entre los diversos métodos de investigación cualitativa propuestos en la bibliografía (la mayoría provenientes del campo de las ciencias sociales), nos centraremos a continuación en Investigación-Acción puesto que es, con mucho, el más utilizado y postulado en Sistemas de Información e Ingeniería del Software. En realidad Investigación-Acción no se refiere a un método de investigación concreto, sino a una clase de métodos que tienen en común las siguientes características (Baskerville, 1999):

- 1) orientación a la acción y al cambio;
- 2) focalización en un problema;
- 3) un modelo de proceso “orgánico” que engloba etapas sistemáticas y algunas veces iterativas; y
- 4) colaboración entre los participantes.

Quizás por no ser un método concreto, existen muchas definiciones de Investigación-Acción. Algunas de las más significativas son:

- Para McTaggart (1991) es “la forma que tienen los grupos de personas de preparar las condiciones necesarias para aprender de sus propias experiencias, y hacer estas experiencias accesibles a otros”.
- Para French y Bell (1996) es "el proceso de recopilar de forma sistemática datos de la investigación acerca de un sistema actual en relación con algún objetivo, meta o necesidad de ese sistema; de alimentar de nuevo con esos datos al sistema; de emprender acciones por medio de variables alternativas seleccionadas dentro del sistema, basándose tanto en los datos como en las hipótesis; y de evaluar los resultados de las acciones, recopilando datos adicionales".
- Para Wadsworth (1998) consiste en la participación de "todas las partes involucradas en la investigación, examinando la situación existente (que sienten como problemática), con los objetivos de cambiarla y mejorarla”.

De las definiciones anteriores se deduce que Investigación-Acción tiene una doble finalidad: generar un beneficio al “cliente” de la investigación y, al mismo tiempo, generar “conocimiento de investigación” relevante (Kock y Lau, 2001). Por tanto, Investigación-Acción es una forma de investigar de carácter colaborativo que busca unir teoría y práctica entre investigadores y practicantes mediante un proceso de naturaleza cíclica. Investigación-Acción está orientado a la producción de nuevo conocimiento útil en la práctica, que se obtiene mediante el cambio y/o búsqueda de soluciones a situaciones reales que le ocurren a un grupo de practicantes (Avison et al, 1999). Esto se consigue gracias a la intervención de un investigador en la realidad del mencionado grupo. Los resultados de esta experiencia deben ser beneficiosos tanto para el investigador como para los practicantes. Una premisa fundamental en esta forma de investigar es que los procesos sociales complejos (y el uso de tecnologías de la información en organizaciones es de este tipo) pueden ser estudiados mejor introduciendo cambios en dichos procesos y observando los efectos de dichos cambios (Baskerville, 1999).

En el campo de los Sistemas de Información, el cliente de una investigación es normalmente una organización a la cual el investigador suministra servicios tales como consultoría, ayuda para cambiar o desarrollo de software, a cambio de tener acceso a datos de interés para la investigación y, en muchos casos, de recibir financiación (Kock y Lau, 2001). En cualquier caso, el investigador que utiliza Investigación-Acción en Sistemas de Información (IA-SI) sirve a dos “señores” diferentes: el cliente de la investigación y la comunidad científica de Sistemas de Información. Las necesidades de ambos suelen ser muy diferentes y, a veces, opuestas entre sí. Intentar satisfacer ambas demandas es el principal desafío que todos los investigadores de IA-SI tienen que enfrentar. La razón de esta supuesta esquizofrenia es que sirviendo tanto las necesidades de los practicantes como las del conocimiento científico, se añaden nuevos elementos a la investigación que hace que sea más deseable.

En un análisis más formal de los participantes en Investigación-Acción, Wadsworth (1998) identifica los siguientes cuatro tipos de roles en este método (en algunas ocasiones la misma persona puede desempeñar más de un rol):

- El *investigador*, el individuo o grupo que lleva a cabo de forma activa el proceso investigador.
- El *objeto investigado*, es decir, el problema a resolver.
- El *grupo crítico de referencia*, aquél para quien se investiga en el sentido de que tiene un problema que necesita ser resuelto y que también participa en el proceso de investigación (aunque menos activamente que el investigador). En él hay tanto personas que saben que están participando en la investigación, como otras que participan sin saberlo.
- El *beneficiario (stakeholder)*, aquél para quien se investiga en el sentido de que puede beneficiarse del resultado de la investigación, aunque no participa directamente en el proceso. Puede ser el receptor de documentos, informes, etc. En este grupo, por ejemplo, caben tanto las empresas que se benefician de un nuevo método para resolver problemas en tecnologías de la información, como los técnicos que aplican dicha metodología.

Desde sus orígenes se han distinguido formas diferentes de aplicar Investigación-Acción (Chein et al, 1948). French y Bell (1996) proponen cuatro variantes que dependen principalmente de las características del proyecto de investigación:

- De *diagnóstico*: el investigador se adentra en una situación problemática, la diagnostica y realiza recomendaciones al grupo crítico de referencia, pero sin que haya un control posterior de sus efectos.
- *Participativa*: el grupo crítico de referencia pone en práctica las recomendaciones realizadas por el investigador, compartiendo con él sus efectos y resultados.
- *Empírica*: el grupo crítico de referencia realiza un registro amplio y sistemático de sus acciones y sus efectos. Esta característica hace que esta variante sea difícilmente aplicable.
- *Experimental*: consiste en evaluar las diferentes opciones que existen para conseguir un objetivo. El principal inconveniente de esta variante reside en la dificultad de poder medir objetivamente las diversas opciones, ya que por lo general serán, o bien aplicadas en distintas organizaciones con distintas características que enturbian los resultados de la investigación, o bien en una sola organización pero en distintos momentos, con lo que el entorno experimental habrá variado.

Un proceso de investigación que emplea Investigación-Acción se halla compuesto de grupos de actividades organizadas formando un ciclo característico. Padak y Padak (1994) identifican los siguientes pasos, que deben seguirse en las investigaciones que utilicen este método:

- 1) *Planificación*: *Identificar las cuestiones relevantes*, que guiarán la investigación, que deben estar directamente relacionadas con el objeto que se está investigando y ser susceptibles de encontrarles respuesta. En esta actividad se buscan caminos alternativos, líneas a seguir o reforzar algo existente. El resultado es que se definen claramente otros problemas o situaciones a tratar. Algunos autores (Baskerville, 1997) distinguen entre diagnóstico (identificar los problemas iniciales) y planificación (especificar acciones para resolver dichos problemas).
- 2) *Acción*: *Variación de la práctica*, cuidadosa, deliberada y controlada. Se efectúa una simulación o prueba de la solución. Es cuando el investigador interviene sobre la realidad.

- 3) *Observación*: *Recoger información*, tomar datos, documentar lo que ocurre. Esta información puede proceder prácticamente de cualquier sitio (bibliografía, medidas, resultados de pruebas, observaciones, entrevistas, documentos, etc.). También se conoce como “evaluación”.
- 4) *Reflexión*: *Compartir y analizar los resultados* con el resto de interesados, de tal manera que se invite al planteamiento de nuevas cuestiones relevantes y, como añade Wadsworth (1998), “a profundizar en la materia que se está investigando para proporcionar conocimientos nuevos que puedan mejorar las prácticas, modificando éstas como parte del propio proceso investigador, para luego volver a investigar sobre estas prácticas una vez modificadas”. También se conoce como “especificación del aprendizaje”. En algunas variantes de Investigación-Acción no es una etapa realmente, sino un proceso continuo que ocurre durante todo el tiempo.

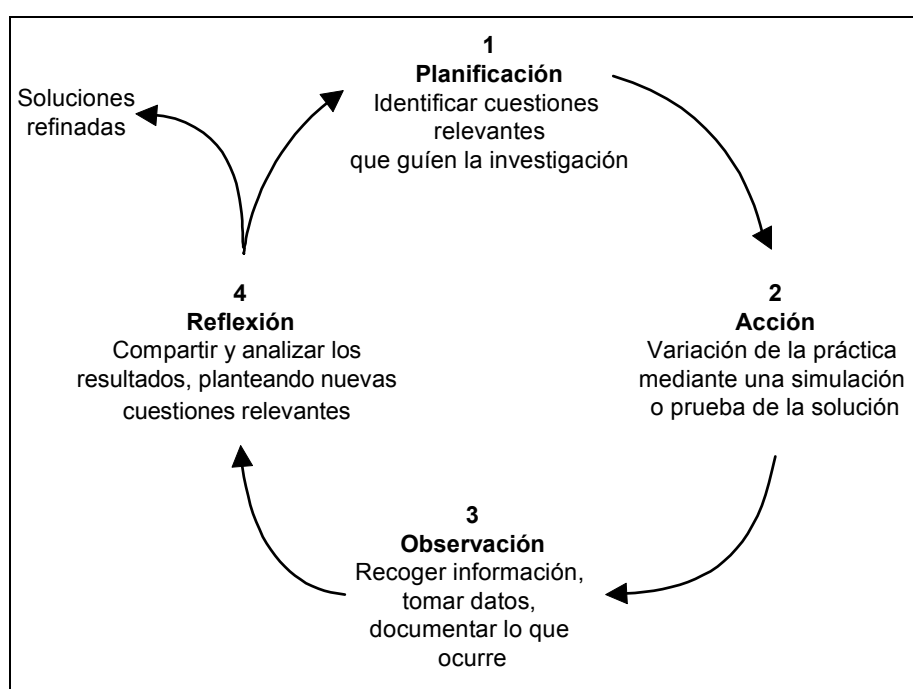


Figura 2-1 . Carácter cíclico de Investigación-Acción.

Con estas características, el proceso definido por Investigación-Acción es iterativo, de forma que se va avanzando en soluciones cada vez más refinadas mediante la compleción de ciclos, en cada uno de los cuales se ponen en marcha nuevas ideas, que son puestas en práctica y comprobadas en el ciclo siguiente, tal como se muestra en la Figura 2-1. Este ciclo caracteriza Investigación-Acción como un proceso reflexivo de aprendizaje y búsqueda de soluciones. El carácter cíclico supone volver a reevaluar o replantear las acciones o caminos a seguir ponderando diagnóstico y reflexión.

En los últimos años Investigación-Acción ha sido reconocido como uno de los métodos de investigación (cualitativa) más potentes en el ámbito de los Sistemas de Información. Ahora bien, la comunidad de especialistas ha detectado diversos problemas en su aplicación que tienen tres causas fundamentales:

- 1) La falta de método con que los investigadores y practicantes utilizan y conciben IA-SI.

- 2) El contexto de consultoría utilizado, que impone una perspectiva demasiado restrictiva al implicar responsabilidades contractuales e intereses organizacionales que pueden ir en contra de lo propuesto por Investigación-Acción.
- 3) La ausencia de un modelo de proceso de investigación definido que indique los pasos a seguir en IA-SI.

Todo lo anterior puede tener como consecuencia una falta de rigurosidad del proceso de investigación. Para solventar estos problemas se han propuesto, entre otras, las siguientes alternativas:

- conducir la investigación usando una perspectiva de gestión de proyectos,
- incluir criterios de calidad especialmente concebidos,
- analizar los factores que inciden en la formalización del proceso, y
- organizar el proceso con una estructura de proyecto.

Combinando varias de estas ideas, Estay y Pastor (2000a, 2000b) han propuesto “usar gestión de proyectos para mejorar el rigor de un proyecto de IA-SI, lo cual se ha traducido en generar una estructura de proyecto que contenga los principales elementos de IA-SI”. Para lograr lo anterior, estos autores plantean la necesidad de adoptar prácticas de gestión, adecuadas a IA-SI, basadas en el PMBOK (*Guide to the Project Management Body of Knowledge*), el modelo de gestión de proyectos -más difundido a nivel internacional- propuesto por el *Project Management Institute* (PMI, 2000). Para Estay y Pastor Investigación-Acción y proyecto son conceptos equivalentes, ya que ambos son experiencias de trabajo únicas con resultados finales igualmente únicos y, además, comparten la idea de intervención, es decir, ambos suponen una alteración voluntaria de la realidad. Aunque la intervención en Investigación-Acción produce alteraciones en una práctica de trabajo, también es una forma de obtener datos de la experiencia real que son necesarios para el proceso de investigación. Los mismos autores también han propuesto un modelo de madurez basado en el *Capability Maturity Model* (CMM), (Paulk et al, 1995), aplicando prácticas de gestión de proyectos de forma incremental con objeto de garantizar una mejora del rigor y calidad del uso de Investigación-Acción en Sistemas de Información (Estay y Pastor, 2001).

En el contexto de la investigación cualitativa en Sistemas de Información se puede considerar que existen dos realidades (científica/académica y práctica) que interactúan pero que se mueven en planos diferentes. IA-SI opera sobre esta realidad dual, que se concreta en dos tipos de ciclos de Investigación-Acción para dos tipos de proyectos:

- a) Ciclos orientados a resolver problemas dentro de proyectos de Sistemas de Información. Estos proyectos consisten en el desarrollo de una solución informática (son proyectos informáticos, de desarrollo de software, de implantación y/o mantenimiento de sistemas informáticos, etc.). En este caso el investigador se encarga de resolver un problema e Investigación-Acción aparece como una herramienta más para el desarrollo de sistemas de información.
- b) Ciclos orientados a investigar dentro de proyectos de investigación. Estos proyectos son esfuerzos intencionados buscando un resultado. En este caso Investigación-Acción nos ofrece un método de trabajo y una justificación para acercarnos a una determinada realidad con fines de probar una teoría o hipótesis.

Por otro lado, la estructura de proyecto de IA-SI propuesta por Estay y Pastor (2000b) define dos ciclos característicos:

- 1) *Ciclo orientado a construir una solución* para generar nuevo conocimiento útil a practicantes y mejorar su práctica. El investigador se conecta con la realidad mediante una intervención. La investigación se utiliza para construir modelos, teorías o conocimiento de manera informada e influida por la realidad. En este ciclo es el interés por resolver un problema lo que origina el interés por la investigación.
- 2) *Ciclo orientado a gestionar la investigación* para producir nuevo conocimiento a la disciplina de Sistemas de Información y mejorar la práctica de los investigadores. En este ciclo es el interés por la investigación el que origina interés por resolver ciertos problemas.

En resumen, IA-SI puede analizarse desde dos dimensiones complementarias (ver Figura 2-2).

- 1) Una dimensión “vertical” en función del tipo de proyecto.
- 2) Una dimensión “horizontal” en función del bi-ciclo típico de la estructura de un proyecto de IA-SI.

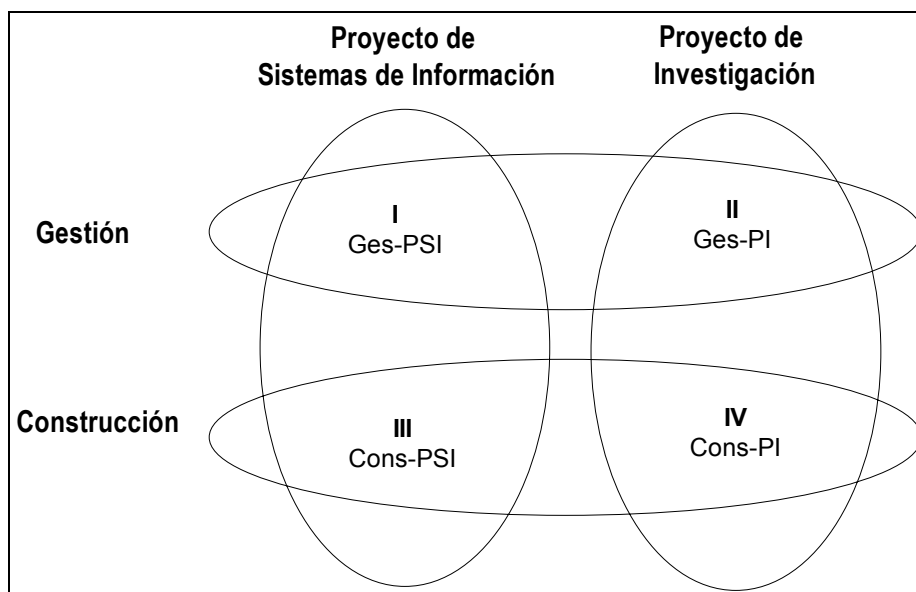


Figura 2-2. Dos dimensiones en Investigación-Acción en Sistemas de Información.

En Lau (1997) se puede encontrar un resumen del uso de IA-SI, comentando diversos ejemplos publicados por diferentes autores referidos a la construcción y desarrollo de sistemas de información, y más especialmente, al análisis, diseño, desarrollo e implementación de software y a los procesos asociados. En Baskerville (1999) se hace una introducción al uso de IA-SI indicando 10 formas de utilización y cuatro características que determinan dicha forma de uso: modelo de proceso (iterativo, reflexivo, lineal); estructura (rigurosa, fluida); rol del investigador (colaborador, facilitador, experto); y objetivos principales (desarrollo organizacional, diseño de sistemas, conocimiento científico, entrenamiento). Baskerville y Wood-Harper (1996) señalan 7 estrategias básicas para llevar a cabo IA-SI: utilizar el “paradigma del cambio”, establecer un acuerdo o contrato formal de investigación, proporcionar

un marco de trabajo teórico, planificar los métodos de captura de datos, mantener la colaboración y el aprendizaje mutuo entre investigador y grupo crítico de referencia, incentivar las iteraciones del ciclo típico, y buscar la generalización de las soluciones.

2.2. Aplicación de Investigación-Acción en este Trabajo.

Teniendo en cuenta que el objeto de investigación en esta tesis es un entorno para la gestión del proceso de mantenimiento del software y que se ha desarrollado en el marco de tres proyectos de I+D en colaboración con dos organizaciones externas, hemos considerado la variante participativa de Investigación-Acción como la más adecuada. A continuación explicamos cómo se ha aplicado dicha variante al desarrollo del trabajo.

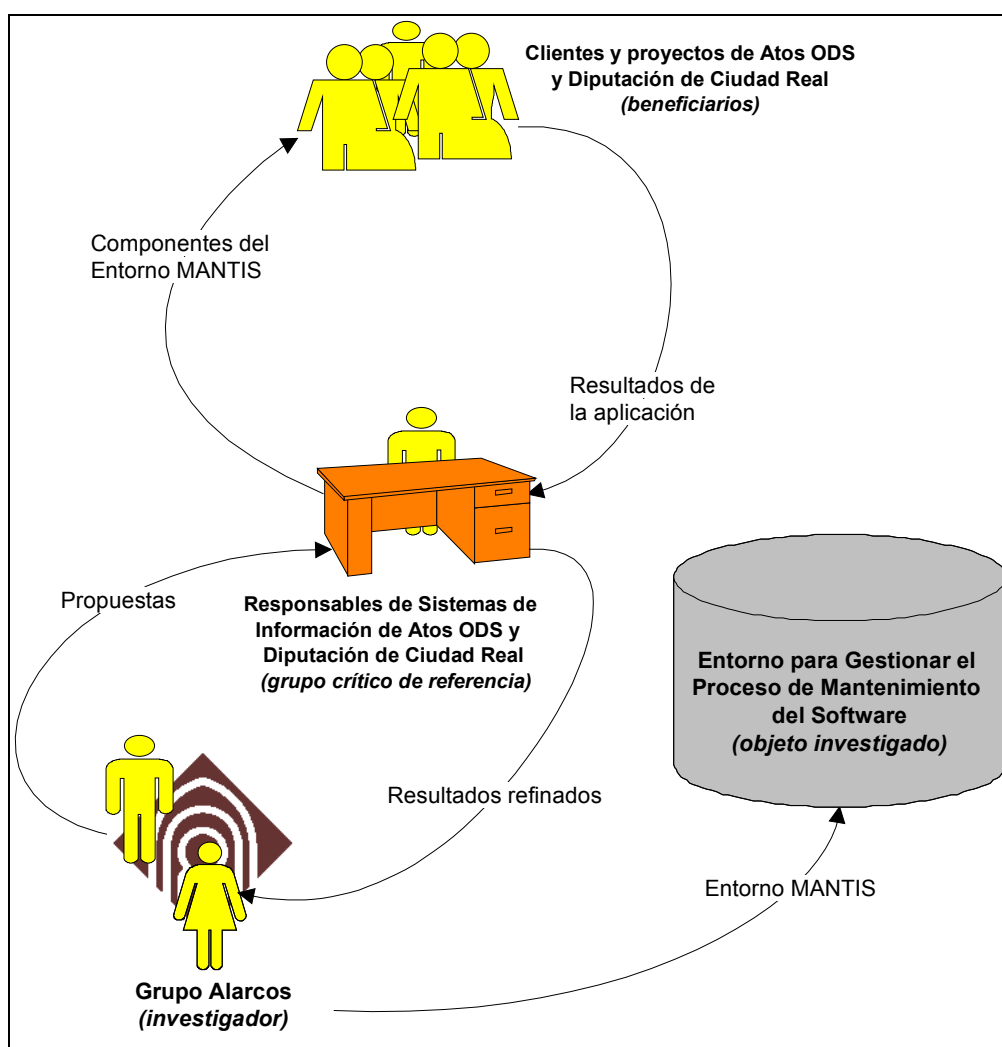


Figura 2-3. Participantes en la aplicación de Investigación-Acción.

2.2.1. Participantes.

En nuestro trabajo, los cuatro roles diferentes ya comentados han correspondido a los siguientes participantes (en la Figura 2-3 ilustramos las relaciones entre ellos):

- a) *Investigador*: el grupo de investigación Alarcos, formado por profesores de la Escuela Superior de Informática de la Universidad de Castilla-La Mancha, en Ciudad Real. El autor de esta tesis es miembro del grupo Alarcos.
- b) *Objeto investigado*: el proceso de mantenimiento del software (PMS), y más en particular, un entorno para gestionar dicho proceso de forma integrada, mejor y más sencilla.
- c) *Grupo crítico de referencia (GCR)*: Ha estado constituido por representantes de las dos empresas que han participado en los proyectos MANTEMA, MPM y MANTIS: Atos ODS S.A. y Exc. Diputación Provincial de Ciudad Real (en realidad el CENPRI, Centro Provincial de Informática).
- d) *Beneficiarios*: Son las organizaciones que pueden ser beneficiadas por los resultados del trabajo, es decir, todas aquellas que poseen software propio o que utilizan software de terceros, que está sometido a un proceso de mantenimiento. Más directamente, los beneficiarios son la Diputación de Ciudad Real (que tiene software propio que mantener) y los clientes que contratan servicios de mantenimiento de software con Atos ODS.

2.2.2. Proceso de Investigación vs Proyectos.

La aplicación de IA-SI más evidente es cuando una organización humana interactúa con sistemas de información. De hecho, Investigación-Acción es de una las pocas aproximaciones válidas para estudiar los efectos de alteraciones específicas en metodologías de desarrollo y mantenimiento de sistemas en organizaciones humanas (Baskerville y Wood-Harper, 1996). Por tanto, la definición de un entorno para la gestión del PMS es un dominio adecuado para la aplicación de Investigación-Acción. Prueba de ello fue que se cumplieron las condiciones idóneas para su uso propuestas por los citados autores:

1. El investigador propuso un marco de trabajo teórico que fue aceptado por el grupo crítico de referencia.
2. El investigador trabajó activamente para que los beneficios fueran mutuos, científicos para el investigador y prácticos para el grupo crítico de referencia.
3. El conocimiento obtenido pudo ser aplicado enseguida.
4. La investigación se desarrolló en un proceso típico cíclico e iterativo combinando teoría y práctica.

La puesta en marcha de Investigación-Acción durante el proceso investigador de este trabajo ha supuesto una continua realimentación entre el investigador y el grupo crítico de referencia: el primero estudiaba los problemas planteados y proponía soluciones que eran analizadas y aplicadas por el segundo en sus ambientes de trabajo real; los resultados eran debatidos en común. De esta forma, con cada ciclo típico (Figura 2-1) llevado a cabo se han ido obteniendo soluciones cada vez más refinadas generadas de forma participativa. Estas soluciones se concretaban en propuestas de componentes del Entorno MANTIS que eran

analizadas y probadas. Los resultados servían para refinar y mejorar los componentes del Entorno o para incluir nuevos componentes.

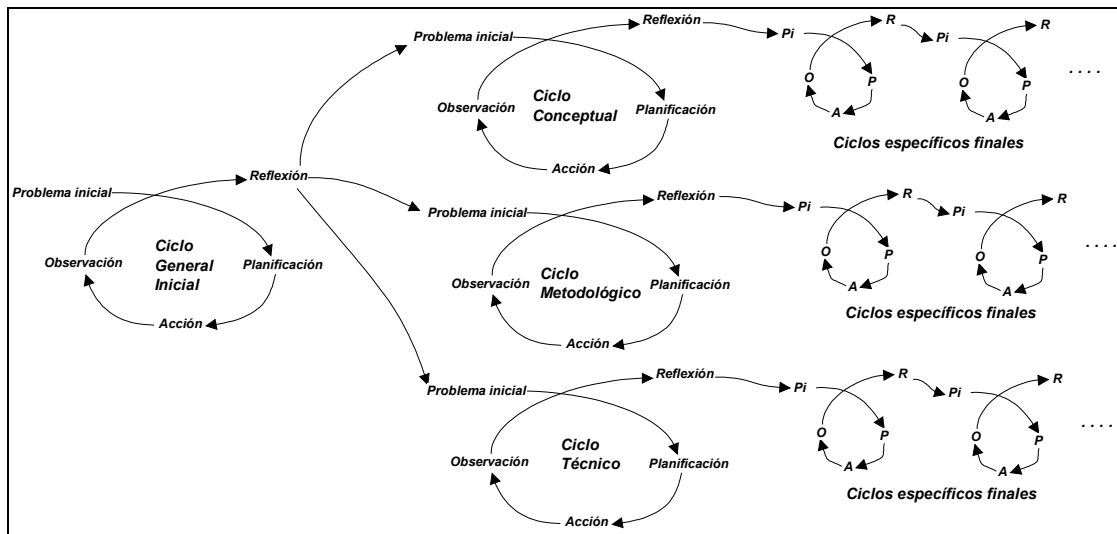


Figura 2-4. Estructura multiciclo con bifurcación utilizada en MANTIS.

Podemos resumir este proceso en los siguientes ciclos (ver Figura 2-4):

- 1) *Ciclo general inicial*: Investigadores y grupo cíclico de referencia definieron la problemática general de la gestión del proceso de mantenimiento del software y establecieron los objetivos y requisitos generales del entorno necesario (planificación). Se procedió a la búsqueda de toda la información de posible interés al respecto (acción). Su análisis posterior (observación) permitió descubrir que el objeto de estudio tenía una complejidad importante dado que debían tenerse en cuenta múltiples aspectos de naturaleza diferente. La compartición y puesta en conjunto (reflexión) entre los investigadores y el grupo crítico de reflexión permitió detectar que las posibles soluciones generales debían consistir en la integración de diversas soluciones parciales a problemas parciales. En resumen, se decidió considerar el Entorno MANTIS como una colección de herramientas de tres tipos diferentes: conceptuales, metodológicas y técnicas (software). Este ciclo supuso un estudio muy amplio de los diferentes aspectos que influyen en la gestión de un proceso software, en general, y del proceso de mantenimiento, en particular: tecnología de proceso software, entornos de ingeniería del software, modelado y metamodelado de procesos, mejora de procesos, arquitecturas conceptuales para modelado de procesos y para integración de herramientas software, modelos de ciclo de vida del software, modelos de gestión de proyectos, etc.
- 2) *Ciclos generales intermedios*: Para cada uno de los tres tipos de herramientas anteriores se realizó un ciclo de Investigación-Acción, con las cuatro etapas conocidas, que pretendía dar respuesta a las siguientes preguntas:
 - a. *Ciclo Conceptual*: ¿Qué se necesita para manejar la complejidad de la gestión del PMS?; ¿Cómo representar toda la información necesaria en un entorno para gestionar el PMS?.
 - b. *Ciclo Metodológico*: ¿Qué métodos/técnicas son útiles para gestionar el PMS?.
 - c. *Ciclo Técnico*: ¿Qué herramientas software son útiles para gestionar el PMS?; ¿Cuáles de ellas interesa desarrollar ex profeso?.

- 3) *Ciclos específicos finales*: A partir del momento en que las respuestas anteriores quedaron claras, tanto para los investigadores como para el grupo crítico de referencia, se procedió a realizar ciclos específicos de Investigación-Acción para cada uno de los componentes individuales del Entorno MANTIS (perteneciente a cualquiera de los tres tipos de herramientas).

Los ciclos anteriores significan que para la definición del Entorno MANTIS se ha utilizado Investigación-Acción con una estructura de proyecto multicíclica con bifurcación (ver Figura 2-4), que es la más conveniente en los casos de nuevos ciclos resultado de nuevos subproblemas y/o cuando se generan nuevos problemas (McNiff, 1988).

La definición del marco de trabajo general para el Entorno MANTIS (con sus tres tipos de herramientas conceptuales, metodológicas y técnicas) facilitó la gestión de la complejidad del trabajo de investigación y ayudó a la incorporación más activa del grupo crítico de referencia, ya que pudieron dedicarse personas diferentes de las dos organizaciones implicadas en diferentes ciclos específicos finales correspondientes a componentes diferentes de MANTIS.

Un problema encontrado en la realización de los diversos ciclos fue la existencia de dependencias no conocidas a priori entre unos ciclos y otros. Por ejemplo, para poderse completar el ciclo metodológico se necesitó llegar a un cierto nivel de refinamiento en el ciclo conceptual. Igualmente, el ciclo técnico necesitó un determinado número de iteraciones en los ciclos conceptual y metodológico. En la realización de los ciclos específicos finales de cada componente del Entorno MANTIS también se descubrieron dependencias que hicieron que el orden temporal en el que fueron avanzando dichos ciclos cambiase continuamente a lo largo del proceso de investigación. Incluso se produjeron algunos casos en que, en mitad del ciclo de un determinado componente, se descubrió la necesidad de añadir a MANTIS un nuevo componente de igual o diferente tipo. Por ejemplo, al realizar una etapa de reflexión en una de las iteraciones del ciclo específico para elegir y definir el componente “arquitectura conceptual” se detectó la necesidad de disponer de una herramienta software que permitiera trabajar con modelos y metamodelos de procesos software de forma integrada a diferentes niveles de abstracción. Al no descubrir ninguna herramienta en el mercado que cumpliera dichos requisitos se decidió incluir un componente técnico nuevo en el Entorno MANTIS para tal fin, llamado MANTIS-Metamod. Estas situaciones supusieron una realimentación de los ciclos intermedios en el sentido de que la realización de sus ciclos específicos finales supusieron con cierta frecuencia una nueva iteración del ciclo intermedio correspondiente.

A lo largo del trabajo de investigación también se han conseguido resultados no previstos inicialmente. Por ejemplo, el reto principal de definir el Entorno MANTIS se abordó en diferentes pasos que, después de diversos refinamientos, permitieron obtener el resultado deseado. La reflexión posterior sobre el camino seguido nos llevó a la conclusión de que habíamos acabado generando una metodología para la definición de entornos de ingeniería del software.

Por otro lado, el trabajo de investigación se realizó en el marco de tres proyectos de I+D que han sido presentados en el capítulo 1. Por esta razón, en el desarrollo de este trabajo nos hemos encontrado con las dos dimensiones mostradas en la Figura 2-2. Por un lado hemos tenido los proyectos MANTEMA, MPM y MANTIS, desde la perspectiva de Sistemas de Información versus el proyecto de investigación propio de esta tesis. Por otro lado hemos tenido el bi-ciclo típico de IA-SI para construir soluciones útiles a los practicantes versus gestionar la investigación para producir nuevo conocimiento útil a los investigadores.

En la Tabla 2-2 se muestra un resumen de los hechos más importantes acontecidos durante la realización de los tres proyectos citados. Se han clasificado según el aspecto al que se refieren (de entre todos los abordados en los citados tres proyectos) y se indica si participaron en su realización el investigador (Grupo Alarcos) y el grupo crítico de referencia (GCR), es decir, Atos ODS y/o Diputación de Ciudad Real. En dicha tabla se puede tener una perspectiva global del trabajo llevado a cabo en estos proyectos. También se puede comprobar que los tres proyectos han sido realmente subproyectos de un proyecto global cuyo objetivo general ha sido mejorar la realización y gestión del proceso de mantenimiento del software (PMS).

Fecha año-mes	Aspecto	Hecho	Roles Participantes	
			GCR	Inves.
1997-07		Inicio del proyecto MANTEMA	X	X
-09	General PMS	Recopilación y estudio de información sobre el PMS		X
-10	Metodología	Planteamiento de una estructura inicial de la metodología		X
-11	Metodología	Reuniones técnicas para refinar la estructura inicial	X	X
-12	Metodología	Propuesta de separar cada tipo de mantenimiento	X	
1998-01	Metodología	Guía para mantenimiento correctivo		X
-02	Metodología	Aplicación de esta guía a proyectos reales	X	
-03	Metodología	Definición de guías para el resto de tipos de mantenimiento		X
-05	Metodología	División del correctivo en urgente y no urgente	X	
-05	Metodología	Presentación de una guía de perfectivo		X
-06	Metodología	Estudio y comentarios a la guía de perfectivo	X	
-06	Metodología	Presentación de MANTEMA 1 (5 guías técnicas)		X
-07	Metodología	Aplicación de MANTEMA 1 a proyectos reales	X	
1999-01	Metodología	Estudio de MANTEMA 1 para corregir defectos y añadir nuevas características		X
-03	Mejora	Búsqueda en la literatura de técnicas de apoyo para la ejecución del proceso		X
-03	Metodología	Construcción de plantillas para los documentos de mantenimiento		X
-04	Metodología	Informe de fallos descubiertos durante la aplicación de MANTEMA 1	X	
-05	Metodología	Petición de integrar los mantenimientos correctivo no urgente, perfectivo, preventivo y adaptativo en un solo tipo (<i>planificable</i>), dejando aparte el correctivo urgente (<i>no planificable</i>)	X	
-06		Inicio del proyecto MPM	X	X
-06	Herr. Técnicas	Prototipo de herramientas para gestionar peticiones de modificación (MANTOOL)	X	X
-07	Metodología	Presentación de MANTEMA 2		X
-09	Metodología	Aplicación de MANTEMA 2 a proyectos reales	X	
-09	Mejora	Estudio de propuestas existentes para la mejora del PMS		X
-10		Fin del proyecto MANTEMA	X	X
-11	Auditoría Control	Propuesta de objetivos de control para auditoría del PMS		X
-12	Gestión Riesgos	Planteamiento de guía para identificar y estimar riesgos		X
2000-01		Inicio del proyecto MANTIS	X	X
-01	Metodología	Refinamiento de MANTEMA 2	X	X

Fecha año-mes	Aspecto	Hecho	Roles Participantes	
			GCR	Inves.
-02	Entorno MANTIS	Estudio de las propuestas sobre entornos de ingeniería del software		X
-02	Gestión Riesgos	Refinamiento de guía para identificar y estimar riesgos	X	X
-03	Metodología	Elaboración de MANTEMA 2.1		X
-04	Metodología	Prueba de MANTEMA 2.1 en proyectos reales de 4GL y PYMES	X	
-04	Entorno MANTIS	Propuesta de “framework” para el Entorno MANTIS con componentes de tres tipos: conceptuales, metodológicos y técnicos		X
-04	Medida	Propuesta de métricas para la estimación del esfuerzo de mantenimiento		X
-05	Entorno MANTIS	Revisión conjunta de los tres tipos de componentes del Entorno	X	X
-05	Medida	Refinamiento de métricas para la estimación del esfuerzo de mantenimiento	X	X
-06	Medida	Propuesta de métricas para la gestión del PMS (reparto de las cargas de trabajo)		X
-07	Arq. Conceptual	Arquitectura conceptual de MANTIS basada en MOF		X
-07	Medida	Refinamiento de métricas para la gestión del PMS	X	X
-09	Ontologías	Integración en MANTIS de la ontología del PMS		X
-09	Flujos de Trabajo	Propuesta de integrar en MANTIS datos sobre la ejecución real de proyectos de mantenimiento	X	
-10	Mejora	Propuesta de cuestionario basado en CMM para la mejora del PMS		X
-11	Flujos de Trabajo	Propuesta de utilizar flujos de trabajo para representar la ejecución de los proyectos		X
-11	Medida	Refinamiento de la propuesta integrada de métricas para el PMS	X	X
-12	Metamodelos	Propuesta de metamodelo de proceso software general para MANTIS		X
-12	Herr. Técnicas	Definición de la arquitectura software del Entorno MANTIS		X
2001-01	Interfaces	Estudio de interfaces metodológicos para integrar los procesos organizacionales y gerenciales (según ISO 15504) en el Entorno MANTIS		X
-01	Flujos de Trabajo	Refinamiento del uso de flujos de trabajo en el Entorno MANTIS (incluir ontología de los flujos de trabajo)	X	X
-02	Roles y actores	Estudio de la integración de aspectos organizativos (roles, organizaciones y responsabilidades) en el Entorno MANTIS		X
-02	Mejora	Prueba del cuestionario para la mejora del PMS en proyectos reales	X	
-03	Ontologías	Ontología de la medida		X
-03	Medida	Propuesta de integración del proceso de medida en el Entorno MANTIS		X
-04	Entorno MANTIS	Propuesta general de Entorno MANTIS		X
-05	Entorno MANTIS	Revisión conjunta de la propuesta de Entorno MANTIS	X	X

Fecha año-mes	Aspecto	Hecho	Roles Participantes	
			GCR	Inves.
-06	Gestión Proyectos	Propuesta de utilizar PMBOK como modelo para la gestión de proyectos		X
-06	Herr. Técnicas	Revisión conjunta de la arquitectura software del Entorno MANTIS	X	X
-06	Mejora	Refinamiento del modelo de mejora del proceso de mantenimiento incluyendo aspectos organizativos y las métricas de gestión y estimación del esfuerzo	X	X
-07	Mejora	Refinamiento del cuestionario para la mejora del PMS	X	X
-07	Herr. Técnicas	Propuesta de Gestor del Repositorio (RepManager) de datos y metadatos basado en XMI		X
-07	Herr. Técnicas	Propuesta de gestor de modelos y metamodelos (Metamod)		X
-07	Interfaces	Refinamiento de la integración de los procesos organizacionales y gerenciales	X	X
-09	Metamodelo PMS	Versión ampliada del metamodelo de proceso software general para MANTIS incluyendo aspectos de medida y los flujos de trabajo		X
-09	Herr. Técnicas	Petición de integrar MANTOOL en el Entorno MANTIS	X	X
-10		Fin del proyecto MPM	X	X
-10	Herr. Técnicas	Prototipo de Gestor del Repositorio (RepManager)		X
-11	Herr. Técnicas	Propuesta de utilizar Sistemas de Gestión de Flujos de Trabajo comerciales para la reificación de los proyectos	X	
-11	Herr. Técnicas	Prototipo de gestor de modelos y metamodelos (Metamod)		X
-11	Herr. Técnicas	Propuesta de interfaz integrado para el Entorno MANTIS: herramienta MANTIS-Tool.		X
-12	Entorno MANTIS	Refinamiento de los componentes del Entorno MANTIS	X	X
-12		Fin del proyecto MANTIS		X

Tabla 2-2. Desarrollo del trabajo en los proyectos MANTEMA, MPM y MANTIS.

Como se puede concluir revisando la tabla anterior, la aplicación del método de Investigación-Acción ha permitido abordar la complejidad del trabajo a desarrollar mediante refinamientos sucesivos, avanzando desde propuestas generales a otras más concretas, llevando a cabo varios ciclos típicos de planificación-acción-observación-reflexión. Además, la utilización de la variante participativa de Investigación-Acción ha supuesto que el grupo crítico de referencia haya jugado un papel cada vez más activo conforme el trabajo ha ido avanzando. De esta forma, el investigador no ha sido visto como un mero consultor o auditor por las organizaciones que formaron el grupo crítico de referencia, que han actuado sintiéndose corresponsables tanto del desarrollo de los proyectos como de los resultados.

En Polo et al (2002a) se explica en detalle cómo se realizó uno de los ciclos específicos, en concreto, el ciclo para definir y construir la metodología MANTEMA para mantenimiento de software. En Ruiz et al (2002d) se han presentado públicamente los principales aspectos, comentados en este capítulo, sobre utilización de IA-SI para la definición del Entorno MANTIS.

2.3. Evaluación de las Herramientas y el Entorno.

Para la evaluación de los componentes del Entorno MANTIS hemos utilizado el método DESMET (Kitchenham, 1996), que es útil para evaluar alguno de los siguientes tipos de objetos:

- Un *método genérico*, es decir, un paradigma para algún aspecto del desarrollo y mantenimiento de software (por ejemplo, diseño estructurado).
- Un *método específico* dentro de un paradigma genérico (por ejemplo, el método de diseño estructurado de Yourdon).
- Una *herramienta*, es decir, una aplicación software que da soporte a una actividad bien definida.

Una colección de métodos o herramientas evaluados de manera conjunta se tratan de la misma forma que métodos específicos y herramientas respectivamente. En cambio, una combinación de diversos métodos y herramientas es tratada como un método si se basan en más de un paradigma genérico, y como herramienta en caso contrario (cuando diferentes herramientas dan soporte a diferentes actividades pero todas ellas consideradas bajo el mismo paradigma genérico). Por las razones anteriores, los componentes conceptuales y metodológicos (de MANTIS o de cualquier otro entorno) deben ser evaluados como métodos específicos y los componentes técnicos como herramientas.

La evaluación de Entornos de Ingeniería del Software (conjuntos integrados de herramientas software) puede ser de dos tipos: se evaluarán como herramientas si todas sus herramientas integradas soportan paradigmas similares, o se evaluarán como métodos diferentes si están basadas en paradigmas diferentes. Según esta última consideración, en la evaluación del Entorno MANTIS de forma general o de subconjuntos parciales del mismo (por ejemplo, los tres tipos de componentes como colección) debe actuarse como en la evaluación de una colección de métodos.

Las evaluaciones con este método son comparativas, es decir, sirven para identificar cual entre varias alternativas es mejor bajo determinadas circunstancias. Las comparaciones también se pueden realizar frente a una idea teórica. Esta última opción ha sido utilizada habitualmente en MANTIS cuando no han existido otras alternativas para comparar. Las evaluaciones con DESMET son dependientes del contexto en que se realizan debido a que no es de esperar que un método/herramienta sea el mejor en todas las circunstancias. Únicamente se producirá una independencia del contexto si se realiza un experimento formal bajo condiciones totalmente controladas, ya que en esta situación sí cabe esperar los mismos resultados independientemente del evaluador y de la organización particulares (Pfleeger, 1994-1995).

El método DESMET distingue dos *tipos de evaluación* principales: cuantitativa y cualitativa. La primera es adecuada cuando el objetivo es establecer efectos medibles de la utilización de un método o herramienta. La segunda es la más idónea cuando el objetivo es determinar la idoneidad de un método o herramienta, es decir, cómo satisface las necesidades de un proceso o la cultura de una organización. Los métodos cuantitativos están basados en la asunción de que es posible identificar alguna propiedad (o propiedades) medible de un producto o proceso software que es posible cambiar como resultado de la utilización del método o herramienta que se quiere evaluar.

Además de la dimensión anterior, en DESMET se distinguen cuatro métodos diferentes de *organizar una evaluación*:

- Como un **experimento formal**: a varios sujetos se les asigna la realización de una o varias tareas usando diferentes métodos o herramientas sujetos a investigación. Las asignaciones entre sujetos y métodos o herramientas se deben realizar de forma tal que los resultados sean imparciales y puedan ser analizados con técnicas estadísticas habituales.
- Como un **caso de estudio**: cada método o herramienta es probado en un proyecto real utilizando los procedimientos habituales de la organización.
- Como una **encuesta**: el personal que ha utilizado un determinado método o herramienta es preguntado sobre dicho método o herramienta. La información obtenida puede ser analizada usando técnicas estadísticas habituales.
- Como un **análisis de características** (*feature analysis*): se identifican los requisitos que los usuarios tienen para una actividad particular y se comparan dichos requisitos con las características que un método o herramienta ofrece para dar soporte a dicha actividad. Este es el tipo de evaluación típica que aparece en las revistas divulgativas de informática comparando paquetes software similares.

Los tres primeros métodos coinciden con los clásicos en investigación cuantitativa en general mientras que el cuarto es típico, junto con el ya comentado de Investigación-Acción, de la investigación cualitativa. En la Tabla 2-3 se muestra un resumen de las diferentes técnicas de evaluación propuestas en el método DESMET en función de las dos dimensiones comentadas.

Técnica	Tipo de Evaluación	Método de Organización	Comentarios
<i>Experimento cuantitativo</i>	Cuantitativa	Experimento	
<i>Caso de estudio cuantitativo</i>	Cuantitativa	Caso de estudio	
<i>Encuesta cuantitativa</i>	Cuantitativa	Encuesta	
<i>Experimento cualitativo</i>	Cualitativa	Experimento	
<i>Caso de estudio cualitativo</i>	Cualitativa	Caso de estudio	
<i>Encuesta cualitativa</i>	Cualitativa	Encuesta	
<i>Pantallazo cualitativo</i>	Cualitativa	Análisis de características	Un individuo realiza una estimación informal de alguna característica
<i>Análisis de efectos cualitativos</i>	Híbrida	Encuesta + Análisis de características	Un experto provee una estimación subjetiva de los efectos cuantitativos
<i>Banco de pruebas (benchmarking)</i>	Híbrida	Experimento + Análisis de características	Se realiza un número de pruebas estándares utilizando métodos o herramientas alternativos y se compara su rendimiento

Tabla 2-3. Diferentes técnicas de evaluación en el método DESMET.

La elección de una o varias de estas técnicas depende de cada caso particular, en función de las siguientes circunstancias:

1. El *contexto de la evaluación* (proyecto individual, selección de estándares para una organización, cambios que forman parte de un programa de mejora, etc.).
2. La *naturaleza del impacto*, que puede ser cuantitativa (por ejemplo, mejora de la productividad) y/o cualitativa (por ejemplo, mejor interoperabilidad de las herramientas).
3. *Naturaleza del objeto evaluado* (método genérico, método específico, o herramienta).

4. *Alcance del impacto*, determinado a su vez por dos factores: la granularidad del producto, que depende de si el método o herramienta se aplica a un producto software completo o a partes individuales (un módulo, un documento, ...); y la extensión del impacto, es decir, las fases del ciclo de vida del producto sobre las que influye el método o herramienta.
5. *Madurez del elemento* (el método o herramientas todavía está en desarrollo, es utilizado en unos pocos productos, o es usado ampliamente en la organización).
6. *Tiempo de aprendizaje* (tiempo requerido para conocer los principios subyacentes y tiempo requerido para ser proficiente en su uso).
7. *Madurez de la organización* para evaluar, que se define en cuatro niveles de capacidad (muy limitada, cualitativa, cualitativa y cuantitativa, y plena).

En MANTIS, las herramientas técnicas (software) han sido evaluadas primero por el investigador actuando de experto (Análisis de Efectos Cualitativos) y después por el grupo crítico de referencia en proyectos reales (utilizando casos de estudio cuantitativos o cualitativos). Cuando no ha sido posible lo anterior, se ha optado por realizar encuestas cuantitativas o cualitativas diseñadas por el investigador y respondidas por miembros del grupo crítico de referencia u otras personas que dicho grupo determinó.

Los actos de la mente, en los que se evidencia su poder más allá de las ideas sencillas, son principalmente tres:

- 1. Combinar varias ideas simples en una nueva compuesta (todas las ideas complejas se conciben así).*
- 2. Combinar dos ideas, simples o complejas, para obtener una visión integrada pero sin llegar a juntarlas, de forma que se tiene una visión de dichas ideas y sus relaciones.*
- 3. Abstracción: separar una idea de todas las demás que le acompañan en su existencia real (todas las ideas generales son concebidas así).*

John Locke en "Un ensayo acerca de la comprensión humana" (1690).

3. Tecnología de Proceso Software

Puesto que el objetivo de esta tesis es la definición de un entorno para la gestión del proceso de mantenimiento del software (PMS), ha sido necesario abordar problemas de diferentes ámbitos de la ingeniería del software (en el capítulo 1 ya se han comentado los temas de SWEBOK que forman el ámbito de esta tesis y las causas de su necesidad). En consecuencia, en este capítulo realizamos una revisión del estado del arte en la tecnología de proceso software, necesaria para dar soporte automático a los procesos de ingeniería del software, ya que, como se verá en el capítulo 5, el entorno MANTIS intenta sacar provecho de los recientes desarrollos en este campo novedoso.

En particular, después de una justificación de la necesidad y ventajas de esta tecnología, se presentan las características generales de los entornos de ingeniería del software (EIS), puesto que MANTIS es una propuesta de esta clase. A continuación, se presentan los conceptos y métodos para el modelado de procesos software, seguido de las principales propuestas internacionales de EIS orientados a procesos. Esto se hace porque la principal característica de MANTIS es su orientación a procesos debido a que su finalidad es dar soporte a uno de ellos, el de mantenimiento. Precisamente, el último apartado aborda los principales modelos de procesos software existentes, que son de utilidad para el mantenimiento.

3.1. Procesos y Tecnologías Software.

Una de las principales líneas de trabajo para la mejora de la calidad de los productos software es el estudio y mejora de los procesos mediante los cuales el software es desarrollado y mantenido. La afirmación anterior está basada en la asunción de que existe una correlación directa entre la calidad del proceso y la calidad del producto obtenido. El área de trabajo, dentro del campo de la Ingeniería del Software, que aborda esta problemática es conocida con el nombre de “Tecnología de Proceso Software” (TPS), o simplemente “Proceso Software”, aunque esta última forma es más utilizada cuando sólo se hace trabajo de investigación sin un desarrollo tecnológico adicional. Como disciplina autónoma, la investigación en la TPS comenzó en los años 80 a través de una serie de eventos (*International Software Process Workshop, European Workshop on Software Process Technology, ...*) y publicaciones (*Journal of Software Process: Improvement and Practice, ...*). La primera contribución importante de la TPS fue la confirmación de que el desarrollo y mantenimiento del software son procesos complejos, que requieren un esfuerzo colectivo y creativo. Por tanto, la calidad de un producto software depende fuertemente de las personas, la organización y los procedimientos utilizados para crearlo, entregarlo y mantenerlo. Un resumen de los resultados obtenidos en la investigación en proceso software puede consultarse en (Derniame et al, 1999a), (Ambriola et al, 1997) y (Cugola y Ghezzi, 1998).

3.1.1. Características de los Procesos Software.

La definición de proceso software complementa el concepto de ciclo de vida³ en el sentido de que éste último define el esqueleto y la filosofía para llevar a cabo un proceso software, pero no es suficiente para guiar y controlar un proyecto de desarrollo y/o mantenimiento. Un proceso software (PS) es “*un conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software*” (Fuggetta, 2000).

La naturaleza especial de los procesos software está determinada por las siguientes características:

- a) *Son complejos.*
- b) *No son procesos de producción típicos;* ya que están dirigidos por excepciones, se ven muy determinados por circunstancias impredecibles, y cada uno tiene peculiaridades que lo distinguen de los demás.
- c) *Tampoco son procesos de ingeniería “pura”;* ya que se desconocen las abstracciones adecuadas (no existe una ciencia experimental en la que apoyarse), dependen demasiado de demasiada gente, el diseño y la producción no están claramente diferenciados, y los presupuestos, calendarios y calidad no pueden ser planificados de forma suficientemente fiable.
- d) *No son (completamente) procesos creativos;* ya que algunas partes pueden ser descritas en detalle y algunos procedimientos han sido impuestos previamente.

³ Un ciclo de vida del software es una definición de los diferentes procesos que afectan a un producto software durante toda su vida (desde su concepción inicial hasta su retirada).

- e) *Están basados en descubrimientos* que dependen de la comunicación, coordinación y cooperación dentro de marcos de trabajo predefinidos: los entregables generan nuevos requerimientos; los costes del cambio del software no suelen reconocerse; y el éxito depende de la implicación del usuario y de la coordinación de muchos roles (ventas, desarrollo técnico, cliente, etc.).

La necesidad de participación humana de forma creativa y la ausencia de acciones repetitivas hacen que ni el desarrollo ni el mantenimiento del software sean procesos de fabricación, pero existen algunas similitudes entre ambos tipos de procesos que son útiles para comprender los procesos software con una perspectiva más amplia. Al igual que los procesos de fabricación, los procesos software constan de dos sub-procesos interrelacionados: el proceso de producción y el proceso de gestión (McLeod, 1990). El proceso de producción se relaciona con la producción y el mantenimiento del producto propiamente dichos, mientras que el proceso de gestión proporciona los recursos necesarios para el proceso de producción y lo controla. Esto último es posible si el proceso de producción devuelve información al proceso de gestión sobre su comportamiento. Estas relaciones están representadas en la Figura 3-1, donde también se muestran las relaciones entre el proceso y el entorno externo: la petición sobre el producto ha de ser llevada a cabo desde el mundo exterior, es decir el entorno exterior es quien justifica la existencia del proceso de producción. Además la gestión tiene que cumplir con los estándares actuales que existen en el entorno, es decir el entorno exterior influye también indirectamente en el proceso de producción. Finalmente, los procesos de producción y de gestión explotan tecnologías que también vienen del entorno.

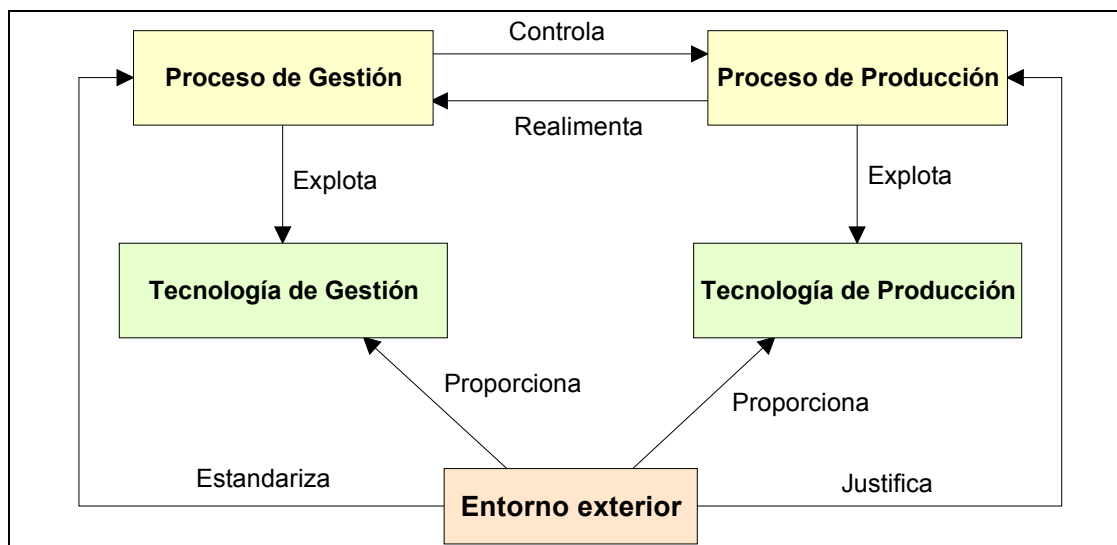


Figura 3-1. Proceso de producción vs proceso de gestión.

3.1.2. Tecnología de Proceso Software.

La esencia de la TPS es que permite la integración de tecnologías de producción y de gestión en un nuevo entorno de trabajo, conocido como “Entorno de Ingeniería del Software orientado al Proceso” (PSEE, *Process-centered Software Engineering Environment* ⁴), que da soporte a los procesos de gestión y de producción de forma integrada. La Figura 3-2 muestra el impacto de esta nueva tecnología, mostrando como el PSEE implementa, controla y mejora los flujos de información con los cuales el proceso de gestión controla al proceso de producción. El objetivo principal de la TPS es dominar la complejidad inherente al proceso software mediante una comprensión profunda del proceso en sí mismo y mediante un soporte automatizado por medio de un PSEE. Un aspecto fundamental para lograr el citado objetivo es el soporte del proceso computerizado, es decir, la disponibilidad de un modelo de procesos y los medios adecuados para definirlo, modificarlo, analizarlo y reificarlo ⁵ (Derniame et al, 1999b).

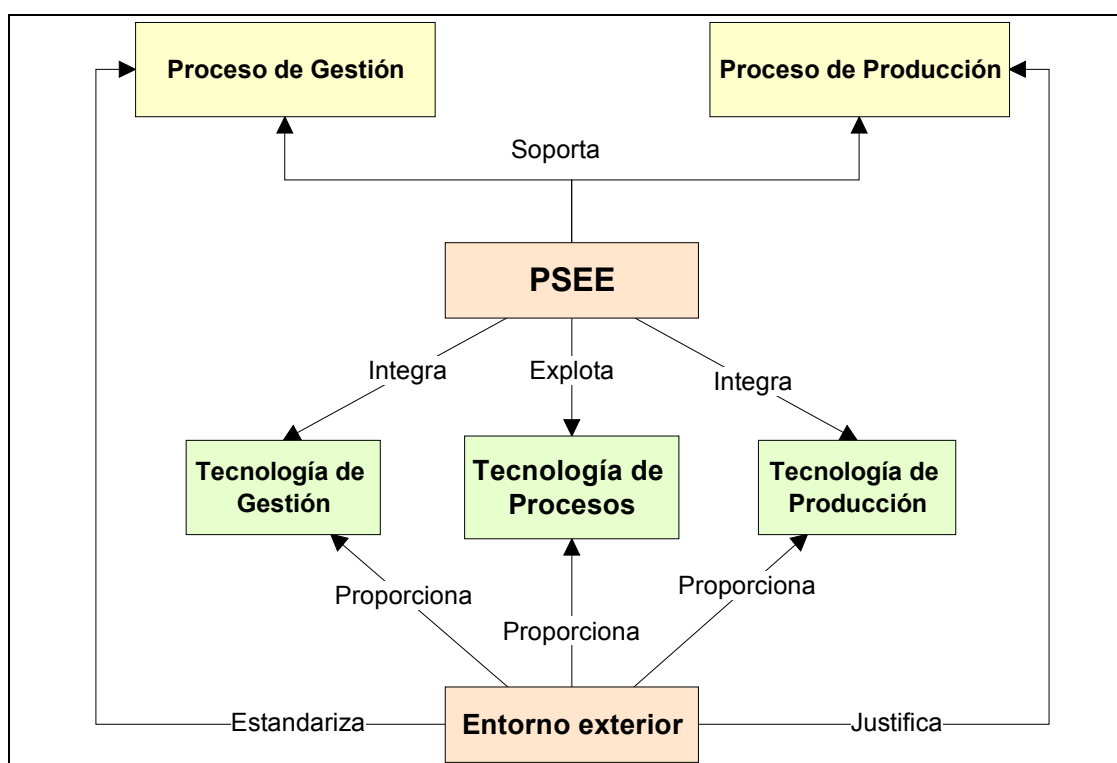


Figura 3-2. Impacto de la Tecnología de Proceso Software.

Consecuentemente con la definición anterior de proceso software, la TPS saca provecho de diversas áreas y conceptos:

⁴ Algunos autores utilizan la nomenclatura alternativa “*Process-sensitive Software Engineering Environment*”.

⁵ Del inglés *enactment*, “transformación de conceptos abstractos en realidades u objetos concretos”. También se utilizan las expresiones animación o ejecución, aunque esta última no se corresponde exactamente con el significado del término. En el DRAE “reificar” es sinónimo de “cosificar”, es decir, “Convertir algo en cosa. Reducir a la condición de cosa aquello que no lo es”.

1. *Tecnologías* de desarrollo y mantenimiento de software: Aportan las herramientas, infraestructuras y entornos que necesitamos para hacer posible y económicamente factible crear y mantener productos software complejos que satisfagan las necesidades sociales actuales y futuras.
2. *Métodos y técnicas* para el desarrollo y mantenimiento de software: Suponen el soporte metodológico esencial para aprovechar de manera eficiente las tecnologías y realizar con éxito las actividades de desarrollo y mantenimiento del software.
3. *Comportamiento organizacional*: la ciencia de las organizaciones y las personas. Útil en TPS porque, en general, los proyectos software se llevan a cabo por equipos de personas que tienen que ser coordinados y dirigidos dentro de una estructura organizacional eficiente.
4. *Marketing y economía*: Los proyectos de desarrollo y mantenimiento de software no son esfuerzos autónomos ya que, como pasa con cualquier otro producto, el software debe estar dirigido a satisfacer las necesidades de clientes reales. Por ello, en los proyectos software son necesarias etapas (especificación de requisitos, etc.) para tener en cuenta adecuadamente el contexto donde se supone que el software será vendido y utilizado.

En conclusión, al desarrollar o mantener software es necesario prestar atención a la compleja interrelación que se produce entre los diversos factores organizacionales, culturales, tecnológicos y económicos.

El lector interesado en profundizar en la TPS puede consultar el GDPA (*Graphical Development Process Assistant*, 2002) un sitio web especializado, que incluye más de 6000 páginas de información sobre este tema, incluyendo conceptos y definiciones, modelos de procesos, entornos, metodologías, elementos de procesos, clases y aproximaciones de modelado de PS, proyectos, herramientas y referencias bibliográficas clasificadas de publicaciones referidas a los puntos anteriores.

3.1.3. Entornos de Ingeniería del Software.

Aunque el uso de herramientas para ayudar a los desarrolladores en la producción de software ha existido, de una u otra manera, desde los días iniciales de la Informática, el concepto de “*Software Engineering Environment*” (SEE), es decir, Entorno de Ingeniería del Software (EIS), es bastante reciente. Un EIS se define como “*una colección de herramientas que proporcionan un soporte automático, parcial o total, a las actividades de ingeniería del software*”. Habitualmente, estas actividades se llevan a cabo en el marco de un proyecto software, y se refieren a aspectos tales como la especificación, desarrollo, reingeniería o mantenimiento de sistemas software. En la bibliografía los EIS también se han conocido con otros nombres (Hanrahan et al, 1994): IPSE (*Integrated Project Support Environment*), ISEE (*Integrated Software Engineering Environment*), coalición de herramientas CASE, herramientas CASE federadas, o ISF (*Integrated Software Factory*).

El término EIS se puede aplicar a sistemas de un alcance muy diferente: desde un conjunto de unas pocas herramientas ejecutándose sobre el mismo sistema, hasta un entorno totalmente integrado capaz de gestionar y controlar todos los datos, procesos y actividades del ciclo de vida de un producto software. Gracias a la automatización de actividades (de forma parcial o total), un EIS puede aportar importantes beneficios a una organización: reducción de costes (alta productividad), mejora en la gestión y mayor calidad en el producto final. Por

ejemplo, la automatización de actividades repetitivas -como la ejecución de casos de prueba- no sólo mejora la productividad, sino que también ayuda a asegurar la completitud y consistencia de las actividades de prueba. Habitualmente, un EIS maneja información relacionada con:

- a) El software en desarrollo o mantenimiento (especificaciones, datos de diseño, código fuente, datos de pruebas, planes de proyecto, ...);
- b) Los recursos del proyecto (costes, recursos informáticos, personal, responsabilidades y obligaciones, ...); y
- c) Los aspectos organizacionales (políticas de la organización, estándares y metodologías empleados, ...).

Un EIS da soporte a actividades humanas mediante una serie de **servicios** que describen las capacidades del entorno. Los servicios proporcionan una correspondencia entre un conjunto de procesos escogidos, relativos al ciclo de vida del software, y su automatización mediante el uso de herramientas. En la mayoría de los casos la funcionalidad de una herramienta está relacionada con uno o más servicios.

El interés investigador en el campo de los EIS comenzó alrededor del año 1990, cuando surgieron las primeras propuestas de modelos de referencia (Zelkowitz, 1993) y se propusieron las primeras clasificaciones de los servicios que deberían incluirse (Zelkowitz, 1996). Pero ha sido a comienzos del siglo XXI cuando se ha empezado a trabajar en propuestas de entornos en la misma línea de la propuesta de esta tesis, es decir, EIS orientados a un dominio específico (Lédeczi et al, 2001) e integrando diversas tecnologías que han sido desarrolladas recientemente (lenguajes de modelado de procesos, metamodelos, XML, etc.). Una prueba de lo anterior se encuentra en la revisión general de la situación de la investigación sobre EIS publicada por Ossher et al (2000). Estos autores indican que las principales líneas de investigación actuales se centran en los siguientes aspectos:

- a) La integración de las herramientas (en las tres áreas de datos, presentación y control), especialmente en base a la utilización de un repositorio que aproveche las potencialidades de XML y tecnologías web complementarias; y
- b) La orientación a procesos, los cuales deben integrar el soporte a herramientas para producir artefactos software con el soporte al modelado y ejecución de los procesos de ingeniería del software que producen dichos artefactos.

3.1.3.1. Integración.

El concepto que más diferencia un EIS de un simple conjunto de herramientas ejecutándose en una computadora bajo un mismo sistema operativo es el grado de integración que el entorno provee. El concepto de integración aplicado a EIS puede significar varias cosas relacionadas pero diferentes:

- El grado en que diferentes herramientas pueden comunicar eficazmente entre sí dentro del marco de trabajo (ver apartado 3.2.2.1) del EIS.
- Una medida de las relaciones entre los componentes de un EIS.
- La facilidad, interoperabilidad, portabilidad, escalabilidad, productividad, etc., producida por la interacción “sin parches” entre un conjunto de componentes de un EIS.

Compartir un mismo sistema de gestión de objetos (gestor del repositorio) en vez de un sistema de ficheros separado para cada herramienta es un aspecto importante de la integración, pero no es el único. Un EIS debe disponer de un conjunto de interfaces que permitan la cooperación arbitraria entre herramientas de fabricantes diversos. Por esta razón, la integración implica los tres aspectos siguientes:

- Un conjunto de servicios. Muchos de los servicios que se describen en apartados posteriores son aplicables a la integración. Por ejemplo, utilizar un sistema de gestión de objetos (SGO) común con esquemas comunes permite que las herramientas compartan objetos; utilizar características de presentación globales en el interfaz de usuario permite disponer de un “aspecto de visualización” similar en todas las herramientas; o los servicios de gestión de procesos y de comunicación son necesarios para que las herramientas puedan comunicarse unas con otras.
- Una nueva dimensión para cada servicio. Tener servicios comunes permite pero no obliga a la integración (los constructores de herramientas no están obligados a utilizarlos). Esta nueva dimensión indica el grado en que un servicio puede contribuir a aumentar la integración.
- Una política. También se requiere implantar políticas para que los constructores de las herramientas, marcos de trabajo y plataformas utilicen los servicios de integración eficientemente. Un ejemplo de esto son las “guías de estilo” para constructores de herramientas.

En base a la definición anterior, la integración es una propiedad de la relación entre los componentes de un EIS. Esta relación puede ser entre herramienta y marco de trabajo, herramienta e interfaz de usuario, herramienta y herramienta, herramienta y SGO, etc.; por eso la integración se puede considerar desde varias **perspectivas** diferentes:

- Para el *usuario*, un EIS integrado provee una vista común del sistema. El entorno entero funciona como una herramienta consistente en vez de como una colección de operaciones diferentes o invocadas separadamente.
- Para el *desarrollador de herramientas*, un EIS integrado provee un interfaz consistente para construir herramientas. De esta manera, las funciones para interactuar con el SGO, gestionar procesos y demás servicios deben estar claramente especificadas; y las herramientas podrán pasar información a otras herramientas de forma fácil.

En general, la necesidad de integración en un EIS abarca varias dimensiones diferentes (Wasserman, 1990):

- **Datos.** La integración de los datos es la habilidad de compartir la información dentro del EIS. El grado de integración de datos puede ser alto (las herramientas usan una base de datos común con un esquema común), mediano (formatos de datos comunes) o bajo (utilizar mecanismos de traducción). Otra característica que la integración de datos puede incluir es la composición.
- **Control.** La integración del control es la habilidad de combinar las funcionalidades ofrecidas en un entorno de forma flexible. Las combinaciones pueden corresponder a preferencias de un proyecto y estar dirigidas por los procesos software subyacentes.
- **Presentación.** La integración de la presentación es la habilidad de interactuar con las funcionalidades del entorno mediante pantallas de apariencia similar y modos de interacción similares.

- **Procesos.** La integración de procesos es la habilidad de acceder a las funcionalidades del entorno utilizando un proceso software reificable predefinido.
- **Marco de Trabajo.** Esto se refiere al grado en que las herramientas están integradas con (es decir, hacen un uso de) el marco de trabajo.

Thomas y Nejme (1992) ampliaron el concepto de integración representado por las dimensiones anteriores (ver Figura 3-3). Estos autores se centraron en las relaciones entre las herramientas y por ello no incluyeron la integración del marco de trabajo, aunque sugieren que esta integración deberá consistir en disponer de mecanismos que mejoren el grado de integración de las otras cuatro dimensiones.

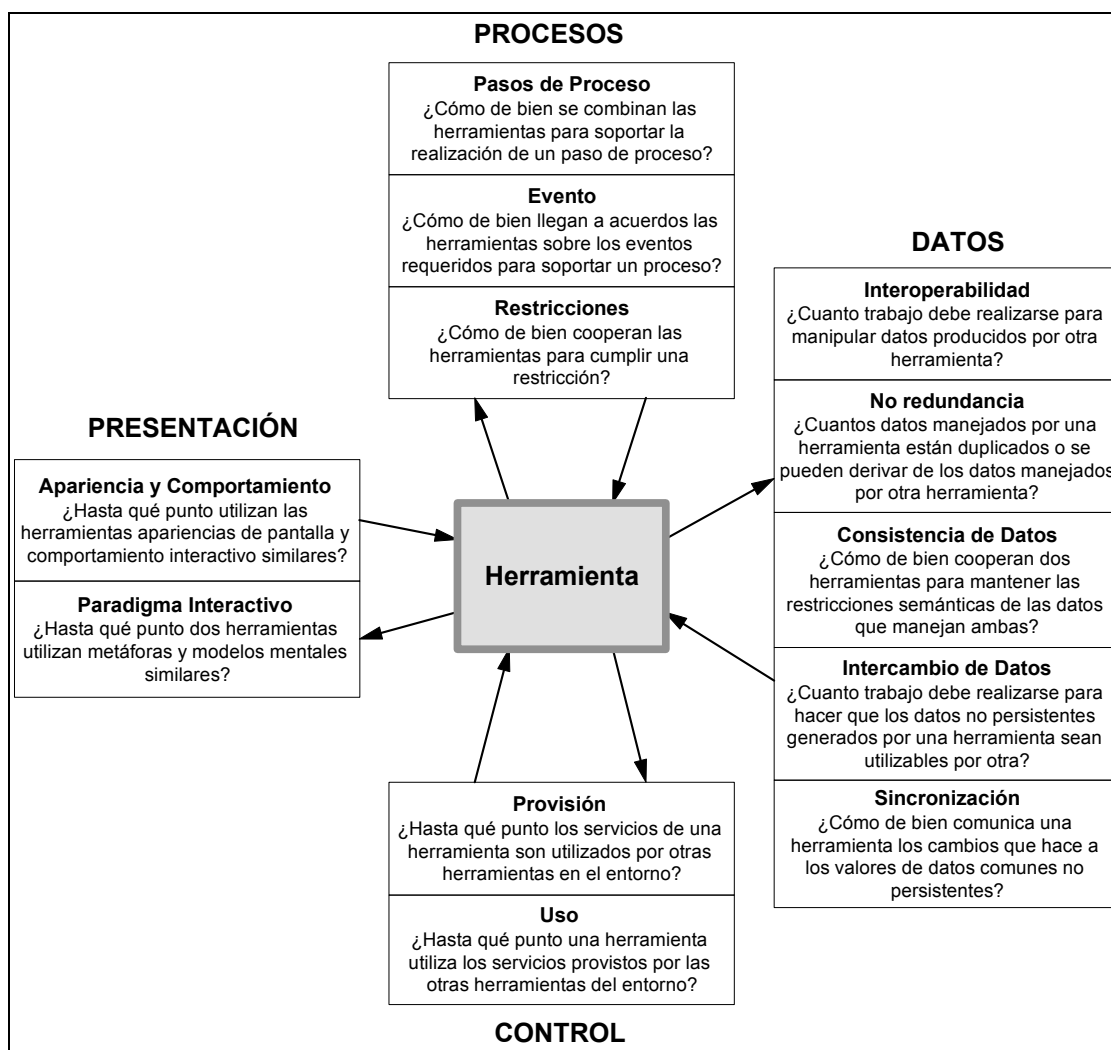


Figura 3-3. Propiedades de la Integración de Herramientas en un EIS.

Las tres primeras dimensiones se pueden representar utilizando un espacio tridimensional (ver Figura 3-4), de forma que la integración que ofrece un determinado EIS se corresponde con un punto discreto del espacio y es posible “visualizar” de forma comparativa varios EIS.

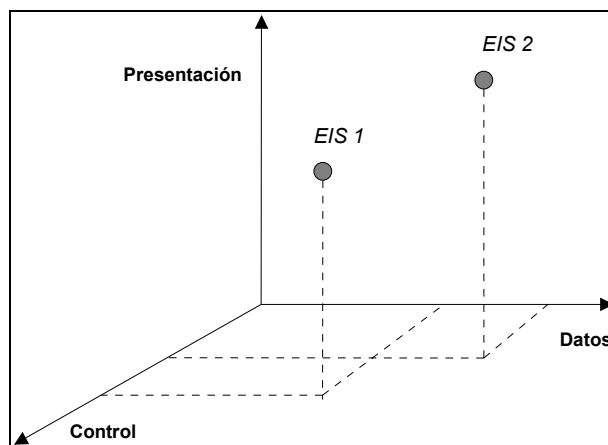


Figura 3-4. Espacio tridimensional de la integración en un EIS.

3.1.3.2. Orientación a Procesos.

Ya se ha comentado la importancia que los EIS orientados a procesos (o PSEE's) tienen en la tecnología de proceso software (ver apartado 3.1.2), de hecho, el principal rol de un EIS es dar soporte para llevar a cabo los PS de forma eficaz. Este punto de vista está ganando peso porque los procesos de desarrollo y mantenimiento de software se han convertido, cada vez más, en actividades complejas y laboriosas de carácter intelectual, con un alto potencial para la mejoras en la calidad y la productividad basadas en la disciplina, la gestión, y la ayuda de EIS y otras tecnologías informáticas. Muchas organizaciones tienen problemas para definir y realizar los pasos que transforman las necesidades del usuario en un producto software de maneras que sean repetibles, medibles con respecto a su impacto en los objetivos de calidad, y adaptables o mejorables. Por tanto, la ayuda de un EIS para implantar un proceso definido durante la realización de un proyecto software puede proporcionar beneficios sustanciales a corto plazo. En este ámbito, un **proyecto** puede ser considerado como una instancia de un proceso que está siendo reificado (Feiler y Humphrey, 1993).

Dentro de un EIS, los servicios de gestión de procesos (ver apartado 3.1.3.3) contribuyen a este soporte eficaz de los PS proveyendo facilidades orientadas al usuario final para definir y utilizar procesos que pueden reemplazar la invocación indisciplinada, difícil de controlar, y tediosa de herramientas individuales. Estos servicios también pueden ayudar a mejorar la arquitectura software de los marcos de trabajo EIS, en el sentido de que los PSEE pueden ser diseñados más fácil y eficazmente (lo que en sí mismo supone un buen ejemplo de mejora de procesos).

Kobialka (1998) ha realizado un estudio en profundidad sobre cómo los procesos software pueden ser soportados por un EIS y, en concreto, cómo implementar el soporte y los cambios de los procesos durante su reificación. Garg y Jazayeri (1996) han considerado que el soporte a procesos en un EIS está basado en las siguientes funcionalidades:

- *Definición de procesos.* Los ingenieros software utilizan el EIS para definir un proceso de cara a su utilización en uno o más proyectos.
- *Análisis de procesos.* Dentro del EIS, un modelo de proceso (ver apartado 3.1.4) puede ser analizado para verificar su consistencia, completitud y corrección.
- *Presentación de procesos.* El EIS incluye soporte para la visualización gráfica de los PS (flujos de actividades) y de los productos (diagramas estructurados).

- *Simulación de procesos.* El EIS soporta el uso de simulaciones para poder evaluar la idoneidad de un proceso antes de consumir recursos en su realización real.
- *Automatización de procesos.* Una vez un proceso ha sido definido, las actividades que no requieren intervención humana pueden ser identificadas y automatizadas por el EIS.
- *Supervisión de procesos.* El EIS supervisa la reificación de un proceso y registra la historia de las actividades realizadas. Esta historia del proceso puede utilizarse después para futuros desarrollos de nuevos procesos o mejora del existente.
- *Soporte de cambios en procesos.* El EIS permite que una organización cambie sus definiciones de procesos sin tener que interrumpir el trabajo (ver apartado 3.1.5).
- *Apertura.* El EIS provee herramientas para intercambiar datos y metadatos con herramientas no integradas o con otros EIS.
- *Soporte multiusuario.* Habitualmente, los proyectos de ingeniería del software son realizados por grupos de personas con diferentes roles, por tanto, el EIS debe dar servicio a todas las personas que trabajan juntas en un proceso.
- *Dirección de procesos.* Los ingenieros software utilizan el EIS para llevar a cabo diferentes etapas de un proceso. El EIS debe ofrecer ayuda para elegir las etapas siguientes en base al modelo del proceso y del estado actual.
- *Interfaz de usuario específico para cada tarea.* Basándose en el modelo del proceso, el EIS puede adaptar el interfaz de usuario a las necesidades de cada tarea y así evitar un exceso de información presentada al usuario.

Los EIS orientados a procesos pueden ofrecer servicios comunes que saquen provecho de la similitud que existe entre los procesos y los productos software: los PS tienen ciclos de vida similares a los ciclos de vida de un producto software porque ambos tienen las siguientes características comunes:

- Pueden ser especificados, diseñados, implementados, reificados (ejecutados), analizados, medidos, modificados y mejorados.
- Pueden tener *arquitecturas de proceso*, que son marcos conceptuales para incorporar, relacionar y adaptar elementos de proceso de forma consistente, incluyendo la capacidad para indicar si un elemento de proceso es o no compatible con la arquitectura. Las arquitecturas de proceso pueden especificar interfaces entre subprocesos, pautas para su composición, y mecanismos de comunicación entre los elementos de proceso. También pueden permitir propiedades orientadas a un dominio de aplicación o a los objetivos de un proyecto concreto, como por ejemplo, el soporte de la reutilización, el prototipado o el desarrollo evolutivo.
- Los *activos de proceso (assets)*, es decir, procesos completos, elementos de proceso, modelos de proceso, arquitecturas de proceso o diseños de proceso, pueden organizarse en bibliotecas y ser reutilizados.

Algunos EIS proporcionan servicios de gestión de procesos utilizando facilidades comunes con otros servicios diferentes. Esto ocurre, por ejemplo, con las facilidades de gestión de objetos, ya que la información de los PS es compleja y además pueden tener que manejar la información del proyecto (así como las representaciones del estado de los procesos) durante la reificación de los procesos.

Cada vez es más frecuente que el desarrollo y mantenimiento de un producto software es realizado con la colaboración de varias empresas u organizaciones. Por ello, en los últimos años ha cobrado auge el estudio de los problemas que surgen cuando se desea que varios PSEE's diferentes y separados colaboren, y más concretamente, que exista interoperabilidad entre los procesos que soportan. Entre las diversas propuestas formuladas para abordar este problema, destacan las federaciones de PSEE's (Martínez et al, 2001a). En esta línea, Ben-Shaul y Kaiser (1998) han propuesto usar la metáfora de la alianza internacional, donde cada organización gestiona sus propios procesos (igual que cada país tiene sus leyes) y los procesos inter-organizacionales actúan de forma semejante al de los tratados entre países. En la bibliografía se han propuesto dos tipos de arquitecturas conceptuales para federaciones de PSEE's: basadas en el control, que favorecen la centralización al existir modelos de procesos comunes; y las basadas en el estado, que disponen de un espacio de trabajo donde se almacena el estado común (Estublier et al, 1998). En (Martínez et al, 2001b) se dispone de una panorámica general sobre el modelado y ejecución de procesos inter-organizacionales.

3.1.3.3. Modelo de Servicios: ISO 15940.

Para describir la relación de posibles servicios que puede soportar un determinado EIS (aunque también se puede aplicar a una única herramienta CASE), vamos a basarnos en la propuesta de estándar ISO 15940 (ISO/IEC, 2001b). Esta norma provee un modelo de referencia y una descripción de todos los servicios que soportan a los procesos del ciclo de vida del software. Un servicio se define como “*una descripción abstracta del trabajo hecho por una o varias herramientas software*”. Cada descripción de un servicio es de carácter general y no asume ningún dominio de aplicación, modelo de ciclo de vida o herramienta en un proyecto. Cada servicio es definido mediante las tres partes siguientes:

- a) *Concepto del servicio*: proporciona una descripción del servicio sin referirse a ninguna implementación concreta.
- b) *Operaciones básicas*: es una lista de las operaciones que deberían ser incluidas en un servicio. La lista no pretende ser exhaustiva y, en muchos casos, incluye sólo conceptos de servicio elementales o primarios.
- c) *Soporte Automatizado*: provee una lista de las operaciones que deben ser automatizadas para cumplir satisfactoriamente con el estándar. Esta lista es un subconjunto de las referidas en el punto anterior.

Los servicios de un EIS se agrupan en seis categorías, que reflejan la gran diversidad y amplitud que tienen hoy en día las diferentes actividades de ingeniería del software. Estas categorías son:

1. Ingeniería Técnica: Soportan actividades -propias de los ingenieros software- relacionadas con la especificación, diseño, implementación, prueba y mantenimiento de software.
2. Gestión Técnica: Dan soporte a actividades mixtas comunes a ingenieros software y gestores de proyectos.
3. Gestión del Proyecto: Ofrecen soporte a actividades relacionadas con la planificación y ejecución de un proyecto software.
4. Gestión de Procesos: Ayudan a los proyectos software a alcanzar disciplina, control y comprensión clara de sus procesos y actividades.

5. Soporte: Útiles a todos los usuarios, están relacionados con procesar, formatear y distribuir datos en formato manejable por las personas.
6. Globales: Ayudan a que la infraestructura del EIS de soporte a las aplicaciones y herramientas. Estos servicios forman el núcleo del marco de trabajo (*framework*) de un EIS.

Servicios de un Entorno de Ingeniería del Software		
Categoría	Servicio	
Ingeniería Técnica	IT.1	Ingeniería de requisitos software
	IT.2	Diseño software
	IT.3	Simulación y modelado software (para determinar la eficacia de diseños alternativos)
	IT.4	Verificación de software
	IT.5	Generación de software basado en componentes
	IT.6	Generación de código fuente
	IT.7	Compilación
	IT.8	Análisis estático de software (análisis de código fuente)
	IT.9	Depuración
	IT.10	Prueba de software
	IT.11	Integración de componentes
	IT.12	Ingeniería inversa de software (capturar información de diseño a partir del código)
	IT.13	Reingeniería de software
	IT.14	Trazabilidad de software (revisar relaciones entre artefactos a lo largo del ciclo de vida)
	IT.15	Pruebas de cualificación de software (comprobar que el producto satisface los requisitos y está listo para su uso)
	IT.16	Prototipado software
	IT.17	Documentación de usuario
Gestión Técnica	GT.1	Gestión de la configuración
	GT.2	Gestión de cambios
	GT.3	Gestión del repositorio EIS
	GT.4	Reutilización (almacenar, inspeccionar y reutilizar activos -assets-)
	GT.5	Colección y análisis de métricas
	GT.6	Aseguramiento de la Calidad
	GT.7	Auditoría
Gestión del Proyecto	GP.1	Planificación
	GP.2	Estimación
	GP.3	Análisis de riesgos
	GP.4	Seguimiento (evolución de costes, tiempos y requisitos)
	GP.5	Evaluación (análisis y toma de decisiones en función de los datos de seguimiento)
Gestión de Procesos	PR.1	Definición de procesos
	PR.2	Biblioteca de procesos
	PR.3	Iniciación de procesos
	PR.4	Utilización de procesos en proyectos
	PR.5	Supervisión de procesos
	PR.6	Mejora de procesos
	PR.7	Documentación de procesos

Servicios de un Entorno de Ingeniería del Software		
Categoría	Servicio	
Soporte	SO.1	Soporte global (crear y manipular objetos dentro del EIS)
	SO.2	Publicación (crear y editar documentos)
	SO.3	Soporte al trabajo en grupo (trabajo cooperativo distribuido)
	SO.4	Soporte a la comunicación de usuarios (facilidad para comunicar con clientes y usuarios)
	SO.5	Administración del EIS (definir y controlar accesos al EIS, controlar su funcionamiento)
	SO.6	Cumplimiento de políticas (implantar políticas de seguridad, accesos, integridad y auditoría)
Globales	GL.1	Gestión de la infraestructura del EIS (gestionar los recursos del EIS)
	GL.2	Comunicación inter-proceso (mecanismo de comunicación básico entre herramientas y/o servicios)
	GL.3	Gestión de objetos (definir, almacenar, mantener y acceder a objetos y sus interrelaciones)

Tabla 3-1. Servicios de un Entorno de Ingeniería del Software.

En la Tabla 3-1 se muestra una lista de todos los servicios EIS incluidos en la propuesta ISO 15940. A continuación se detallan los servicios que están dentro del alcance de este trabajo, es decir, aquellos que se corresponden con las categorías de procesos de gestión y de organización en la norma ISO 15504-2 (ver Figura 3-20). En esta lista se incluyen dos servicios de gestión técnica, todos los servicios de gestión del proyecto y de gestión de procesos, y un servicio de soporte (están marcados en negrita en la Tabla 3-1).

3.1.3.3.1. Servicios de Gestión Técnica.

GT.4 - Reutilización:

- a) Concepto: Ofrece soporte para el almacenamiento, inspección y reutilización de activos de proceso (*assets*).
- b) Operaciones básicas:
 - Depositar, conseguir o enviar activos al repositorio.
 - Catalogar, registrar y clasificar los activos.
 - Buscar o revisar en el repositorio.
 - Examinar o extraer los activos.
 - Registrar el extractor o el remitente.
 - Informar de los requisitos de uso y limitaciones de un activo.
- c) Automatización:
 - Registro y catalogación de activos.
 - Registrar remitente, fecha, limitaciones y requisitos de uso de un activo.
 - Búsqueda por funcionalidad en el repositorio.
 - Hojear activos del repositorio.

GT.5 - Colección y análisis de métricas:

- a) Concepto: Provee facilidades para la recolección y organización de datos primitivos en forma de información de interés para los usuarios del EIS.
- b) Operaciones básicas:
 - Insertar y borrar datos de un conjunto de datos (*data set*).
 - Escoger un modelo de medición adecuado para un conjunto de datos determinado.
 - Comparar un conjunto de datos con lo previsto por el modelo.
 - Calcular estadísticas generales sobre un conjunto de datos.
- c) Automatización:
 - Recolección de datos primitivos.
 - Elegir modelo de medición.
 - Crear la funcionalidad de un nuevo modelo.
 - Importación y exportación de conjuntos de datos.
 - Realización de cálculos estadísticos sobre un conjunto de datos.
 - Elaboración de gráficos sobre un conjunto de datos.
 - Análisis de regresión.

3.1.3.3.2. Servicios de Gestión del Proyecto.

GP.1 - Planificación:

- a) Concepto: Provee operaciones para manejar datos relativos a los objetivos y restricciones relevantes de un proyecto.
- b) Operaciones básicas:
 - Descomposición de objetivos de proyecto en eventos clave (hitos).
 - Cuantificar entradas y salidas de los paquetes de trabajo.
 - Calcular plazos de espera de los eventos.
 - Calcular fechas de inicio y finalización.
 - Analizar caminos críticos.
 - Generar calendarios detallados de eventos.
 - Permitir la asignación de recursos.
 - Establecer esquemas de control del proyecto.
 - Registrar la asignación de responsabilidades de trabajo a individuos u organizaciones.
 - Establecer canales de comunicación entre los miembros del equipo del proyecto.
 - Establecer criterios de evaluación.
- c) Automatización:
 - Representación gráfica de los hitos frente al tiempo.
 - Nivelado de recursos.
 - Análisis de caminos críticos.
 - Análisis de impacto de la creación/modificación/eliminación de tareas.
 - Cálculos y previsiones presupuestarias.

GP.2 - Estimación:

- a) Concepto: Soporte para cuantificar, analizar y predecir los costes y recursos necesarios en el proyecto.

- b) Operaciones básicas:
 - Crear y modificar estimaciones de coste, tamaño y recursos.
 - Mantener un histórico de estimaciones de coste, tamaño y recursos.
 - Generar estimaciones para parámetros variables.
- c) Automatización:
 - Cambiar estimaciones cuando sea requerido por cambios en los requisitos.
 - Análisis de sensibilidad frente a los parámetros variables.

GP.3 – Análisis de riesgos:

- a) Concepto: Soporte a las actividades de planificación y evaluación de los elementos que tienen relación con eventos negativos para el proyecto.
- b) Operaciones básicas:
 - Realizar análisis “que pasaría si” probando los efectos de retrasos en la disponibilidad de recursos o en el calendario del proyecto.
 - Hacer el análisis de los riesgos en costes, recursos, tiempos y fallos.
 - Estimar costes y probabilidad de fracaso debido un análisis inadecuado de los requisitos de usuario.
 - Generar informes sobre diversas estrategias de asignación de recursos.
- c) Automatización:
 - Medición de costes y tiempos.

GP.4 – Seguimiento:

- a) Concepto: Ofrece soporte para el seguimiento del progreso del proyecto (costes, tiempos, requisitos de usuario, etc.).
- b) Operaciones básicas:
 - Recopilar métricas relativas al estado actual de un proyecto y sus paquetes de trabajo.
 - Comparar estimaciones de costes, tamaño y recursos con los valores actuales.
 - Mostrar el estado de todas las variables de un proyecto, y producir datos e información de resumen del proyecto.
 - Hacer un seguimiento de los requisitos de usuario.
- c) Automatización:
 - Disponibilidad de métricas adecuadas.
 - Análisis de tendencias en las desviaciones de las estimaciones de costes, tamaño y recursos frente a los valores actuales.
 - Activar alarmas cuando los datos actuales de utilización de recursos difieran significativamente de los previstos; o cuando cierta acción no acabe en un determinado periodo de tiempo.

GP.5 – Evaluación:

- a) Concepto: Soporta el análisis, evaluación y toma de decisiones asociados con los datos de seguimiento, métricas obtenidas y criterios de aceptación de los usuarios.

- b) Operaciones básicas:
 - Comparar datos de costes y tiempos con datos de seguimiento y con los criterios de evaluación del proyecto.
 - Crear, modificar y almacenar comentarios de usuarios, recomendaciones y errores software.
 - Evocar la aceptación del usuario para cada requisito.
 - Colectar métricas para hacer análisis “que pasaría si” y análisis de impacto.
- c) Automatización:
 - Evaluación de los resultados de proyecto/producto frente a los criterios de aceptación del usuario.
 - Análisis de impacto para ayudar a determinar un camino de recuperación con impacto mínimo.

3.1.3.3.3. Servicios de Gestión de Procesos.

PR.1 - Definición de procesos:

- a) Concepto: Ayuda a la implantación de los procesos organizacionales, cubriendo el ciclo de vida del software a través de la adaptación y particularización de un conjunto de clases de procesos de referencia de alto nivel. Esta adaptación incluye las políticas específicas de la organización en cuanto a sus infraestructuras, métodos y procedimientos.
- b) Operaciones básicas:
 - Analizar los requisitos de proceso, incluyendo los específicos del dominio y los específicos de la aplicación.
 - Instanciar, componer, descomponer, particularizar y modularizar definiciones de proceso.
 - Simular, modelar y validar definiciones de proceso.
- c) Automatización:
 - Todo lo anterior.

PR.2 - Biblioteca de procesos:

- a) Concepto: Ofrece soporte para reutilizar capacidades de procesos en base a sus activos o “*assets*” (actividades, tareas, etc.). Un activo de proceso puede ser tan sencillo como la definición de una tarea simple o tan complejo como la definición de un ciclo de vida completo. Los activos de proceso pueden ser objetos versionados (con diferentes versiones).
- b) Operaciones básicas:
 - Crear, modificar y eliminar activos de proceso.
 - Certificar, medir y administrar activos de proceso.
- c) Automatización:
 - Almacenamiento y versionado de activos de proceso.
 - Procesamiento de informes de estado.

PR.3 - Iniciación de procesos:

- a) Concepto: Soporta la asignación de un modelo de ciclo de vida, un conjunto de procesos y el EIS para satisfacer los requisitos y restricciones de un proyecto particular.

- b) Operaciones básicas:
 - Revisar criterios y restricciones de un proyecto y seleccionar modelo de ciclo de vida.
 - Definir relaciones y particularizar procesos y actividades.
- c) Automatización:
 - Definición de interrelaciones y particularización de procesos y actividades.

PR.4 - Utilización de procesos en proyectos:

- a) Concepto: Incluye capacidades para ayudar a utilizar procesos dentro de un proyecto (p.ej., asignación de usuarios, facilidades navegacionales, etc.).
- b) Operaciones básicas:
 - Ofrecer ayuda sobre el proceso y proveer orientación a los miembros del equipo del proyecto.
 - Consultar e informar sobre utilización y estado de procesos.
 - Especificar, recolectar e informar sobre métricas de procesos.
 - Interactuar con simulaciones de definiciones de proceso y representaciones de alto nivel.
- c) Automatización:
 - Consulta e información sobre la utilización y estado de los procesos.

PR.5 - Supervisión de procesos:

- a) Concepto: Soporta la observación, detección, registro y traza de actividades de procesos (dentro de proyectos).
- b) Operaciones básicas:
 - Establecer condiciones y criterios de supervisión.
 - Observar la evolución en el estado de procesos reificados.
 - Detectar la ocurrencia de eventos de proceso específicos.
 - Registrar la ocurrencia de eventos de proceso específicos.
- c) Automatización:
 - Detección y registro de la supervisión.
 - Presentación de datos de supervisión, incluidos gráficos si es necesario.
 - Distribución de datos de supervisión.

PR.6 - Mejora de procesos:

- a) Concepto: Soporta la evaluación, medición y modificación de los procesos específicos organizacionales y de proyectos, y de los ciclos de vida de proyectos.
- b) Operaciones básicas:
 - Definir objetivos de eficiencia.
 - Identificar mediciones relacionadas con los objetivos.
 - Establecer valores límites para la consecución de objetivos.
 - Evaluar la capacidad de los procesos.
 - Preparar informes de evaluación comparando los datos actuales con los deseados.
 - Planificar las evaluaciones.
- c) Automatización:
 - Recolección de datos de mediciones.
 - Preparación de informes de evaluación.

PR.7 - Documentación de procesos:

- a) Concepto: Da soporte a todos los demás servicios para la documentación del proceso.
- b) Operaciones básicas:
 - Identificar los requisitos de documentación.
 - Diseñar y elaborar documentos de procesos.
 - Producir y editar documentos de procesos.
 - Distribuir documentos de procesos.
 - Mantener documentos de procesos.
- c) Automatización:
 - Diseño, producción y edición de la documentación..
 - Distribución y mantenimiento de la documentación.

3.1.3.3.4. Servicios de Soporte.

SO.6 – Cumplimiento de políticas:

- a) Concepto: Ayuda a implantar políticas de seguridad, control de acceso, integridad de datos y objetos, intercambio seguro de datos y objetos, y auditoría del proceso.
- b) Operaciones básicas:
 - Establecer información de seguridad: identificar usuarios y asociarles privilegios de acceso adecuados.
 - Gobernar el acceso a los contenidos de información.
 - Proteger datos y objetos de modificaciones no autorizadas.
 - Exportar e importar datos y objetos de forma segura.
 - Hacer control y auditoría de la seguridad..
- c) Automatización:
 - Identificación y autenticación.
 - Control de acceso obligatorio y discrecional.
 - Protección de la integridad.
 - Seguridad en la exportación e importación.

3.1.4. Modelos de Proceso Software.

En una organización o en un dominio de aplicación, los procesos de diferentes proyectos tienden a seguir patrones comunes, o bien porque la mejores prácticas son formalmente reconocidas o bien por la existencia de estándares. Por lo tanto, se hace necesario intentar capturar estos aspectos comunes en una representación del proceso, la cual describe estas características comunes y fomenta la homogeneidad.

Durante las tres últimas décadas, el estudio de los procesos de producción del software ha llevado al desarrollo de varios *Ciclos de Vida* del software que pueden ser empleados en las actividades de ingeniería del software. Estos son, por ejemplo, los modelos en Cascada, Evolutivo, Transformacional o en Espiral. Las funciones primarias de un ciclo de vida son determinar las fases envueltas en el desarrollo y mantenimiento del software, y establecer los criterios de transición para la progresión de una fase a la siguiente. Estos últimos incluyen a los

criterios de compleción para la fase actual y a los criterios de entrada para la siguiente. Estos modelos de ciclo de vida ayudan a los ingenieros y a los gestores a comprender mejor el proceso software, y a determinar el orden de actividades globales envueltas en la producción de software. Sin embargo, una limitación de estos modelos es que no dan mayor importancia a procesos que son cruciales para el éxito de proyectos software. Por ejemplo:

- La *Gestión de la Configuración* es crítica, sin embargo ¿cómo debe hacerse la gestión de la configuración para que un producto X sea integrado correctamente en un proceso específico en el proyecto Y?
- Identificar medidas y métricas de acuerdo al nivel de calidad requerido y a las políticas de control definidas es fundamental para obtener productos software de alta calidad. Pero, ¿Cómo son integradas estas medidas y métricas en el proceso de producción y mantenimiento de software?
- La Gestión de Proyectos es esencial para el éxito de un proyecto. ¿Cómo puede ser recogida y presentada la información al jefe del proyecto para que tenga una visión global del proceso actual?
- Durante el mantenimiento, todas las actividades que van desde “Control de Cambios” a “Corrección de Errores” han de estar bien coordinadas. ¿Cómo se garantiza la consistencia cuando dos “correctores” trabajan simultáneamente en la misma versión de una configuración?

Como ya se comentó, el objetivo último de la TPS es alcanzar el punto donde la representación de procesos puede ser usada para conducir los actuales procesos de desarrollo y mantenimiento del software. Como primer paso en esa dirección, la tecnología de procesos introduce la noción de **Modelo de Proceso (MP)**, una representación abstracta de una familia de procesos expresada en una adecuada notación de modelado de procesos (formalismo). De esta forma, la TPS hace uso de la considerable experiencia existente con representaciones formales y construcción de modelos. La utilidad de los modelos de procesos es múltiple: desde un simple soporte informal (ayuda a incrementar la comprensibilidad) hasta la asistencia directa en la evaluación y mejora de los procesos.

Para facilitar la comprensión de los PS, Murer et al (1996) proponen utilizar una perspectiva tridimensional al consultar y “visualizar” los MP; con dos dimensiones estructurales (dependencias de objetos y flujos de trabajo concurrentes) y otra dimensión de evolución del proceso. La primera dimensión se refiere a que un PS y su ciclo de vida pueden representarse como un conjunto parcialmente ordenado de objetos (contratos, especificaciones, implementaciones, documentos, código, pruebas, etc.). La segunda dimensión tiene que ver con las transiciones de estado de los objetos (con restricciones indicando las permitidas). La tercera dimensión se refiere a poder ver la historia del proceso, incluidas las diferentes versiones que han existido de cada objeto.

En general, la disponibilidad de un MP computerizado (es decir, representable y manipulable mediante un computador) proporciona capacidades adicionales para:

- *Completar procesos*: dando soporte directo a los desarrolladores para el control de su trabajo, su coordinación con otros, etc.;
- *Automatizarlos*: permitiendo la invocación automática de herramientas no interactivas;
- *Dirigirlos*: facilitando el soporte indirecto, mediante la información del estado actual del proceso, el significado de los puntos de decisión, etc.

- *Incrementar su eficiencia:* puesto que un MP preciso es primordial en el aumento de la efectividad, ya que proporciona una base no ambigua para la comunicación entre los procesos.

Los MP también juegan un rol esencial en diversas actividades de ingeniería del software, entre las que merece destacar las siguientes:

- *Seguimiento:* porque los MP permiten una clara comprensión de lo que puede ser observado y por qué;
- *Simulación:* porque el comportamiento de un proceso puede ser estudiado al menor coste sin desarrolladores reales o herramientas.
- *Validación:* porque la simulación y supervisión, junto con la inspección de modelos, permiten que las propiedades de los modelos sean determinadas.
- *Verificación:* porque los MP son necesarios para probar formalmente las propiedades de interés del proceso.
- *Mejora:* porque todas las actividades anteriores son necesarias para la mejora de procesos.

El modelado de procesos puede entenderse desde dos puntos de vista distintos (McChesney, 1995): Un modelo descriptivo describe cómo un proceso es realizado en un entorno particular; y un modelo prescriptivo describe cómo un proceso debería ser realizado.

Los MP descriptivos tienen que ser exactos, es decir, deben describir fielmente los procesos reales. Una organización implicada en un proceso de mejora necesita utilizar ambos tipos de modelos: un modelo descriptivo para representar los procesos actuales a ser evaluados (actividad necesaria para mejorarlos); y un modelo prescriptivo para representar los procesos mejorados que se tienen como objetivo a alcanzar. Becker-Kornstaedt et al (1997) proponen un aproximación al modelado descriptivo de procesos software basada en 8 etapas que se muestra resumida en la Figura 3-5.

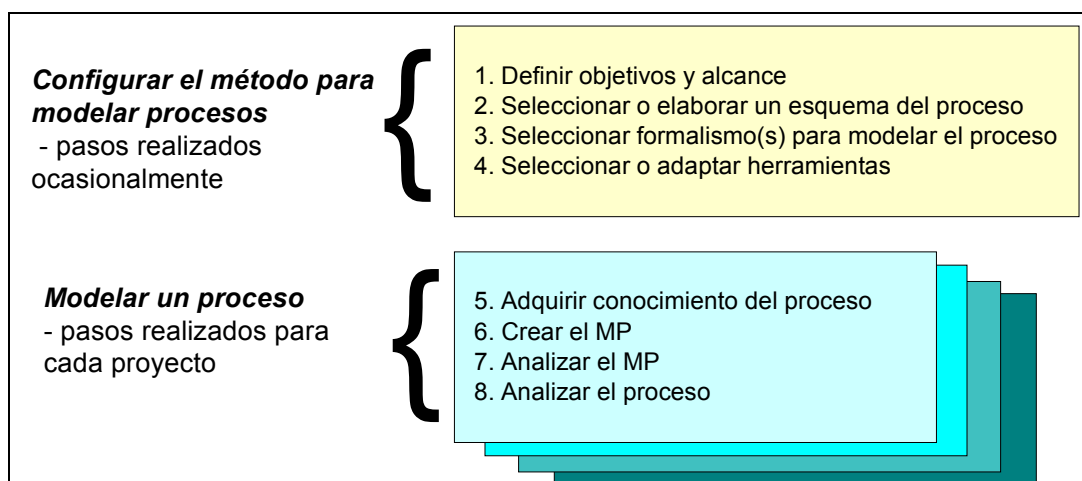


Figura 3-5. Método para el modelado descriptivo de procesos.

McChesney (1995) divide los MP prescriptivos en dos categorías: manuales y automatizados. Los MP manuales pueden ser estándares, metodologías y métodos centrados en los procesos del ciclo de vida del software (gestión, desarrollo, mantenimiento, evaluación, soporte organizacional, etc.). Los MP automatizados son especificaciones computerizadas de estándares de PS. Su principal objetivo es actuar como guía en el proceso de modelado, es decir, ayudar a los agentes de proceso para interpretar mecánicamente los MP (Lonchamp, 1994). Los MP automatizados se subdividen en dos categorías: orientados a actividades (centrados en las funciones, actividades y partes de los PS) y orientados a personas (centrados en la especificación de las personas participantes y sus relaciones). En Acuña y Ferré (2001) se puede consultar una taxonomía completa de los MP prescriptivos, con ejemplos de cada una de las categorías y subcategorías.

Los tipos de información en un MP también pueden estructurarse desde diferentes puntos de vista. En este sentido, Curtis et al (1992) establecieron las siguientes perspectivas de información:

- *Funcional*: que representa los elementos de proceso que están siendo implementados y los flujos de información que son importantes para dichos elementos.
- *De Comportamiento*: que representa cuando y bajo qué condiciones son implementados los elementos de proceso.
- *Organizacional*: que representa donde y por quién (en una organización) son implementados los elementos de proceso.
- *Informativa*: que representa las entidades de información resultantes o manipuladas por un proceso, incluyendo su estructura y relaciones.

3.1.4.1. Elementos Básicos.

Cualquier MP debe incluir un conjunto mínimo de elementos o conceptos. En ingeniería del software se considera que dichos componentes mínimos son los siguientes (Derniame et al, 1994); (ver Figura 3-6) :

- Una **actividad** es una operación atómica o compuesta, o un paso de un proceso. Las actividades se encargan de generar o modificar un determinado conjunto de **artefactos**; para lo cual siguen o están basadas en unos determinados procedimientos, reglas y políticas (**normas**). Además, una actividad es un concepto con un componente funcional fuerte ya que implica entradas, salidas, y resultados intermedios.
- El conjunto de artefactos a ser desarrollados, entregados y mantenidos en un proyecto es lo que se denomina el **producto**. Normalmente existe algún tipo de correspondencia entre la descomposición de actividades en sub-actividades y del producto en sub-productos, pero no es obligatorio.
- Un **recurso** es un activo que una actividad necesita para poder ser llevada a cabo. En ingeniería del software hay dos clases principales de recursos: los **desarrolladores**, es decir, los **agentes** humanos que intervienen en el proceso; y las **herramientas**, es decir, los agentes computerizados (software) que tradicionalmente han sido usados en desarrollo y mantenimiento del software (herramientas CASE como editores especializados, compiladores, etc.) o herramientas de propósito general como hojas de cálculo, editores de diagramas, etc., que pueden ser utilizados para gestionar el proceso.

- Normalmente, las herramientas tienen una relación fuerte con las actividades en las que son usadas, mientras que los desarrolladores están vinculados indirectamente con una actividad por medio de sus **roles**, es decir, el conjunto de responsabilidades, obligaciones y tareas (por ejemplo, diseñador, jefe de proyecto, revisor, etc.). El carácter de la **organización** impacta en el proceso indirectamente por medio de los roles definidos, y directamente por medio de las **normas** (políticas, reglas, y procedimientos) que gobiernan las actividades. Las normas suelen aparecer en forma de documentos, y por tanto, suele existir una jerarquización entre normas y sub-normas. La TPS ayuda a hacer cumplir estas directrices proporcionando soporte computacional para guiar y controlar las iniciativas de los desarrolladores, para coordinar sus actividades, o para soportar la cooperación y/o automatización. De esta forma, la tecnología de procesos contribuye a aumentar el nivel de confianza en la calidad del producto, ya que asegura mayor adherencia a los estándares de calidad de procesos.

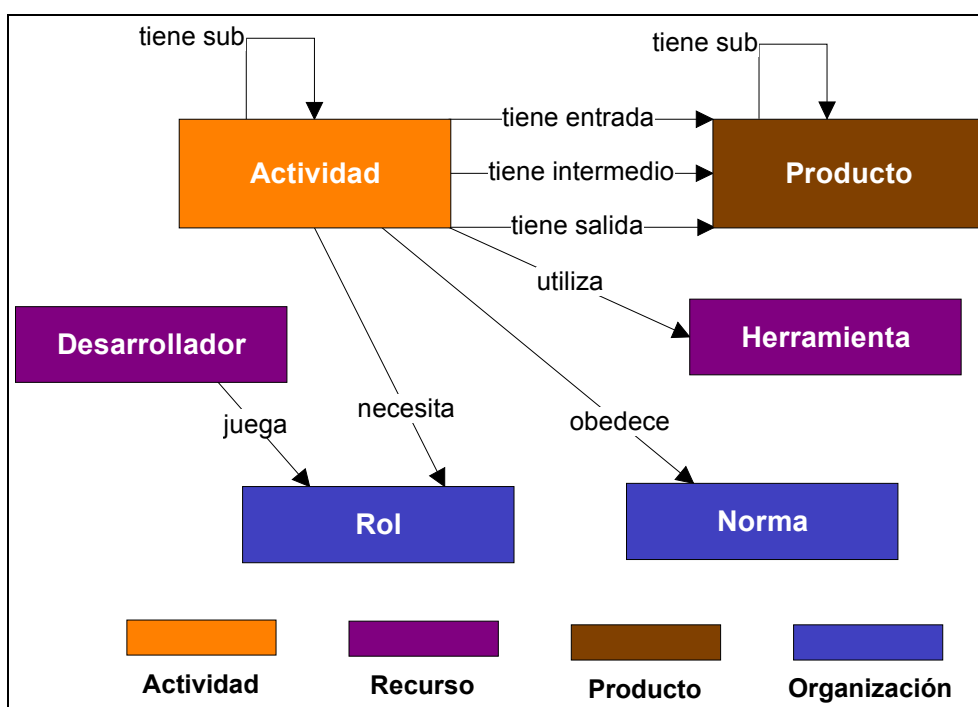


Figura 3-6. Elementos básicos de un Modelo de Proceso.

No todos los autores consideran el mismo conjunto básico de elementos de proceso, aunque existe consenso en cuanto a considerar los siguientes: Agentes/Actores/Desarrolladores, Roles, Actividades, y Artefactos/Productos (Dowson et al, 1991), (Feiler y Humphrey, 1993). En (Acuña y Ferré, 2001) se puede consultar una comparativa de las principales propuestas.

3.1.4.2. Niveles y Dominios.

Los procesos pueden ser representados con diferentes niveles de detalle, capturando clases cada vez más pequeñas de sub-procesos, correspondientes a asuntos cada vez más detallados. Normalmente, las descripciones más genéricas, como los ciclos de vida, suelen ser informales o semi-formales, mientras que en el otro extremo los procesos operacionales de soporte suelen necesitar una descripción formal en un nivel bastante detallado. En función de

este nivel de detalle o abstracción, en la bibliografía han sido establecidas las siguientes distinciones:

- Un ciclo de vida es una representación general e informal de procesos software. El propósito de estos modelos es dirigir aspectos metodológicos globales. Por ejemplo, resaltar que se debe especificar antes de codificar (modelo en cascada; Royce, 1993), o que se deben calcular los riesgos antes de la especificación (modelo en espiral; Boehm, 1981), o ir desde la especificación al código y no al contrario (modelo transformacional; Balzer et al, 1983).
- Un MP genérico es una representación abstracta, que puede ser usada en muchos proyectos similares y organizaciones que comparten propiedades y características comunes. Por ejemplo, puede ser una representación formal de un ciclo de vida o de un proceso de reingeniería del software.
- Un MP adaptado (*customized*) es bastante más detallado que un MP genérico. Normalmente se deriva de un modelo genérico teniendo en cuenta características específicas locales, es decir, dominios de aplicación. En esta clase se incluyen modelos con diferentes niveles de detalle, por ejemplo: un estándar internacional, adaptación (*tayloring*) de un estándar a las necesidades de una organización, o particularización para ajustarse a las necesidades de proyectos específicos.
- Un MP reificable (*enactable*) tiene el mayor nivel de detalle, de forma que define el proceso concreto y exacto que debe ser seguido en un proyecto específico. Desde la perspectiva de la tecnología de procesos software, un MP reificable es una representación lista para ser cargada en una máquina y proporcionar soporte automático a la gente que lleva a cabo procesos de desarrollo o mantenimiento de software.
- Finalmente, un MP reificado (*enacting*) es un modelo que da soporte a un proyecto real y concreto, para lo cual incluye algún tipo de representación del estado actual del proceso.

De forma ortogonal a la anterior clasificación, se distinguen dos dominios de aplicación en cada uno de dichos niveles (Dowson y Fernström, 1994). El dominio de realización de procesos se refiere al acto de participar las personas y las herramientas en un proceso. El dominio de reificación de procesos está relacionado con el acto de conducir automáticamente un proceso, es decir, interpretar con más detalle el modelo del proceso. En el ámbito de la tecnología de procesos software, las herramientas y las personas también pertenecen al dominio de reificación de procesos ya que son controladas por el entorno. En otras palabras, mientras que un proceso se lleva a cabo en el mundo real, es reificado en el mundo abstracto del modelo computerizado, pero puesto que los computadores son parte del mundo real, se dice que la reificación es parte de la realización.

Es precisamente por la pertenencia a ambos dominios por lo que, contrariamente a lo que se pudiera pensar, los desarrolladores humanos pueden contribuir creativamente en el desarrollo y mantenimiento de software, incluso cuando se trabaja bajo las restricciones de la reificación en un PSEE. Esta multiplicidad de dominios y la idea de múltiples vistas, son aspectos inherentes a la naturaleza compleja de los procesos software.

3.1.4.3. Vistas.

De forma similar al concepto de vistas en el campo de las bases de datos, las vistas de un MP expresan un punto de interés particular en vez del MP completo. Ejemplos típicos son:

- El *modelo de actividades*, que se centra en los tipos, estructura y propiedades de las actividades del proceso y sus relaciones. Esta vista es relevante, por ejemplo, al jefe de proyecto para propósitos de organización del calendario.
- El *modelo de productos*, que describe los tipos, estructura y propiedades de los elementos software de un proceso. Además de para los miembros del equipo del proyecto, esta vista puede ser de interés también para el usuario, por ejemplo, para comprender qué tipo de documentación se elaborará.
- El *modelo de recursos*, que describe los recursos que se necesitan o se suministran a los procesos, que es relevante desde una perspectiva de gestión de proyectos (costes, calendarios, riesgos).
- El *modelo de roles*, que describe un conjunto de recursos peculiar, como son las habilidades que los desarrolladores poseen y las responsabilidades que aceptan. Esto es relevante, además de para el equipo de desarrollo del proyecto, para la organización responsable y, en particular, para el personal de aseguramiento de la calidad.

Obviamente, una vista no puede ser definida sin usar conceptos de otras. Por ejemplo, una vista de actividades puede necesitar algunos elementos de una vista de productos para que sea comprensible, o bien, una vista de productos puede llevar información sobre la estructura de un producto, reduciendo la actividades a meras entradas y salidas.

Cabe destacar que hay dos tendencias con respecto a como utilizar la idea de diferentes vistas en un MP comprensivo. La primera, con una perspectiva *top-down* inspirada en la experiencia con la gestión de bases de datos, considera las vistas como sub-modelos de un modelo único global y obliga a la consistencia de las diferentes vistas. La segunda aproximación, de tipo *bottom-up*, asume como punto de partida una colección de vistas (posiblemente inconsistentes), que el diseñador puede combinar negociando las inconsistencias mediante un soporte automático (Nuseibeh et al, 1993).

3.1.4.4. Sistemas de Procesos Software.

Algunos autores establecen una clasificación en niveles de los procesos de ingeniería del software. En este sentido, Wang y King (2000) establecen la siguiente taxonomía (en orden de menor a mayor jerarquía):

- *Práctica*: Una actividad o un estado en un PS para llevar a cabo una tarea específica. Es la unidad mínima que puede ser modelada.
- *Proceso*: Es un PS tal como ya se ha definido. Expresado en función del concepto anterior, un proceso es un conjunto de prácticas funcionalmente coherentes y reutilizables para un proyecto software.
- *Categoría*: Es un conjunto de procesos funcionalmente coherentes y reutilizables en algún aspecto de la ingeniería del software.
- *Subsistema*: Es un conjunto de categorías funcionalmente coherentes y reutilizables en alguna parte principal de la ingeniería del software.
- *Sistema*: Conjunto completo de PS estructurados.

En la Tabla 3-2 se muestra un resumen de algunos de los sistemas de PS más conocidos, indicando entre corchetes el número de elementos de cada nivel que incluye. Por ejemplo, en el nivel de procesos, CMM, Bootstrap e ISO 15504 incluyen 18, 23 y 35 procesos, respectivamente.

Nivel	CMM (Paulk et al, 1993)	Bootstrap (Koch, 1993)	ISO 15504 (ISO/IEC, 1998c)
Sistema	CMM	Bootstrap	ISO 15504-2
Subsistemas	-	Áreas de proceso [3]	-
Categorías	Niveles de proceso [5]	Categorías de proceso [9]	Categorías de proceso (PC's) [5]
Procesos	Áreas de Prácticas Clave (KPA's) [18]	Procesos [32]	Procesos (PR's) [35]
Prácticas	Prácticas Clave (KP's) [150]	Atributos de Calidad de Sistema (QSA's) [201]	Prácticas Base (BP's) [201]

Tabla 3-2. Taxonomía de los principales sistemas de procesos software.

Complementario a la taxonomía anterior, los miembros de un sistema de PS se pueden clasificar según una jerarquía funcional. Wang y King (2000) proponen la jerarquía mostrada en la Figura 3-7, donde un sistema de PS se puede analizar desde tres puntos de vista: el modelo de procesos, el modelo de evaluación, y el modelo de mejora. A su vez, el modelo de procesos se subdivide en tres subsistemas: organización, producción (desarrollo y mantenimiento), y gestión.

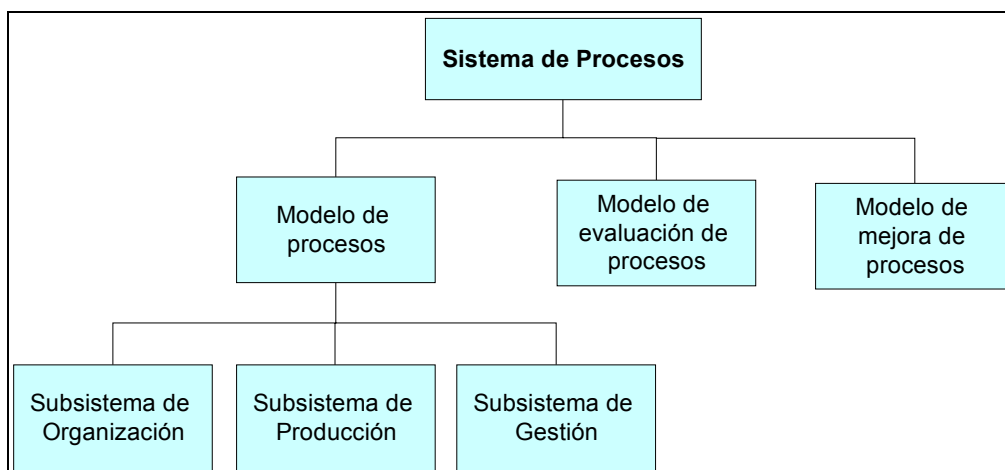


Figura 3-7. Jerarquía funcional en un sistema de procesos software.

Un buen ejemplo de la jerarquía representada en la Figura 3-7 es el conjunto de modelos de procesos de ingeniería del software definido en los estándares ISO 12207 e ISO 15504 (ver apartados 3.3.1 y 3.3.2).

3.1.5. Evolución de Procesos Software.

En un entorno de producción de software se deben considerar dos aspectos:

1. Un proceso, P, es decir, el proceso de producción del mundo real que incluye actores humanos, herramientas CASE, etc.; y que sirve de guía para llevar a cabo los actividades de desarrollo y/o mantenimiento de productos software, y
2. Un modelo de procesos, MP, que es una representación del mundo real, y captura el estado actual de las actividades para dirigir, hacer cumplir o automatizar partes del proceso de desarrollo o de mantenimiento.

Idealmente, P y MP deberían estar perfectamente alineados en el sentido de que el estado interno del modelo de proceso debe ser una fiel representación del estado actual de los asuntos en el mundo real, es decir, “*P es una instancia de MP*”. Sin embargo, cualquier proceso software en el mundo real es un proceso creativo y dinámico que abarca numerosos actores humanos y no puede ser reducido a la mera programación de autómatas. Puesto que un grado de inconsistencia entre P y MP puede traer consecuencias negativas para la calidad final del producto, el modelo de procesos debe adaptarse a cualquier evolución del proceso en el mundo real, además de coordinar las actividades y gestionar los flujos de información. Son varias las razones por las que puede cambiar un proceso software (Madhavji, 1991):

- ser erróneo;
- ciertos pasos importantes no están previstos, lo que hace ineficaz al proceso;
- el MP puede ser genérico y necesita ser detallado para obtener resultados específicos;
- las presunciones sobre las cuales se construyó el MP ya no son válidas; ó
- las dinámicas políticas, humanas y tecnológicas pueden inducir el cambio.

Como resultado de lo anterior, el proceso P del mundo real y el modelo de procesos MP deberían evolucionar de forma conjunta y coherente; aunque esta afirmación no siempre se puede cumplir; En esta línea, Heimbigner (1993) establece dos clases de cambios:

- *Cambios Definicionales*: que ocurren cuando se detecta un error o inadecuación en el MP. El problema es modificar el MP y a continuación propagar los cambios a todas las instancias de ejecución (procesos P del mundo real) de dicho MP.
- *Cambios en Instancias*: que ocurren cuando, por razones imprevistas, el proceso real P debe ser alterado aunque el MP sigue considerándose válido (por ejemplo, una revisión de diseño no se lleva a cabo por problemas de tiempo). La ocurrencia repetida del mismo cambio en varias instancias puede implicar la necesidad de un cambio definicional. En este ámbito, Kobialka (1998) ha realizado una propuesta muy completa para dar soporte al cambio de procesos durante su reificación.

Como conclusión general de lo anterior, se puede decir que la evolución de un proceso software es, en sí misma, un proceso completo de alto nivel, conocido como **meta-proceso**, cuyo principal objetivo es asegurar que P y MP son consistentes entre sí. Este meta-proceso incluye los pasos para cambiar los procesos del mundo real y los pasos para introducir cambios en el MP. En consecuencia, para llevarlo a cabo correctamente, el director del proyecto necesitará implicar los servicios del modelador de procesos para diseñar un modelo aumentado,

validar este nuevo modelo, y después decidir cuando comenzar a reificarlo. Conradi y Jaccheri (1999) proponen las siguientes fases en el ciclo de vida del meta-proceso:

1. Elicitación y especificación de requisitos del proceso.
2. Análisis del proceso.
3. Diseño del proceso.
4. Implementación del proceso.
5. Reificación del proceso.
6. Evaluación de la calidad y rendimiento del proceso.

Para profundizar en este tema, se dispone de una buena presentación de los diferentes aspectos relacionados con el meta-proceso en (Conradi, 1994), especialmente en cuanto a los conceptos, técnicas y métodos para evolución de procesos. Además, Nguyen y Conradi (1994) establecen una taxonomía de los diversos tipos de meta-procesos software. También puede ser interesante consultar en (Cunin, 2000) la presentación del proyecto ESPRIT PIE (*Process Instance Evolution*), dedicado al estudio de la evolución de los PS en circunstancias de participación humana intensiva. En concreto, PIE propone incluir en los PSEE diversos servicios avanzados de supervisión y soporte a la toma de decisiones para la evolución dinámica de PS, es decir, para los cambios que tienen lugar durante su ejecución (Alloui et al, 2000).

3.1.6. Lenguajes de Modelado de Procesos.

Un lenguaje de modelado de procesos (LMP) software permite expresar los procesos software en base a un determinado modelo de procesos utilizando una determinada sintaxis. Para ello, es necesario poder describir todos los elementos de proceso (actividades, roles, recursos, etc.) y los elementos del meta-proceso (relacionados con la evolución del proceso).

Son 6 las clases de elementos de proceso (elementos primarios) que debe poder modelar un LMP: actividades, productos, roles, personas, herramientas, y soporte para la evolución. Los cinco primeros ya han sido comentados (ver Figura 3-6). En cuanto al último (ver apartado 3.1.5), se concreta en que el LMP debe ofrecer soporte para la evolución del MP, tanto a nivel técnico (por ejemplo, mediante la reflexión o interpretación) como a nivel conceptual (mediante un metamodelo asociado). Además de estos elementos primarios o principales, los LMP suelen incluir la posibilidad de representar elementos propios del meta-proceso (elementos secundarios). Los más habituales son:

- Proyectos/Organizaciones: Una organización está formada por personas relacionadas con otras personas y elementos de otros tipos. Un proyecto es una estructura temporal de una organización montada para poder alcanzar un objetivo específico.
- Contextos de Trabajo: Están formados por espacios de trabajo, cada uno de los cuales engloba y controla artefactos para un (sub)proceso. Estos artefactos suelen ser ficheros de un repositorio.
- Vistas de Usuario: Son el interfaz general para ayudar al usuario a comprender el MP y guiarlo durante su reificación. A este nivel se distingue entre el llamado “modelo interno” (que muestra cómo funciona el proceso) y el “modelo externo” (que muestra cómo se hacen las cosas).

- Modelo de Cooperación: En el se indican los modos de cooperación (secuencial o paralelo), los protocolos de comunicación entre objetos, y la coordinación de acciones (ordenación o sincronización).
- Modelo de Versionado/Transacciones.
- Modelo de Calidad/Rendimiento.
- Modelo de Calidad del Producto: Representa los objetivos de calidad del producto y las métricas asociadas.
- Modelo de Rendimiento del Proceso: Para expresar el cumplimiento con respecto a tiempos, costes, roles, etc.

Otro aspecto que va a influir en las características y naturaleza de los LMP es que pueden ser utilizados con propósitos diversos (Fugetta, 2000):

- *Comprender el PS*: Un LMP puede ser utilizado para representar de forma precisa cómo está estructurado y organizado un proceso. Esto es una ayuda para eliminar inconsistencias en la especificación del proceso.
- *Diseñar el PS*: Proactivamente, un LMP puede usarse para diseñar un nuevo PS, describiendo su estructura y organización.
- *Entrenamiento y formación*: Una descripción precisa del PS puede ser útil para enseñar los procedimientos y operaciones de una empresa al personal recientemente incorporado.
- *Simulación y optimización del PS*: Una descripción de un PS puede ser simulada para evaluar los posibles problemas, cuellos de botella, y oportunidades de mejora.
- *Soporte al PS reificado*: Una descripción precisa de un PS puede ser interpretada por una computadora y, por tanto, utilizada para proveer diferentes niveles de soporte a las personas que llevan a cabo el proceso.

Por otro lado, un MP es desarrollado, analizado, refinado, transformado y reificado dentro del meta-proceso. Por tanto, además de modelar adecuadamente el proceso del mundo real, un LMP se usa de diferente forma por diferentes roles durante las diferentes fases del ciclo de vida del meta-proceso. En consecuencia, en cada fase interesan unas características diferentes en el LMP. Por ejemplo, durante la “Especificación de requisitos” interesa un lenguaje orientado al modelado conceptual, intuitivo y con notación fácil para los usuarios no técnicos (por ejemplo, basado en gráficos). En cambio, durante la “Implementación del proceso” el lenguaje debe permitir el suficiente detalle para que el MP sea reificado, es decir, el LMP debe ser ejecutable (formal). En general, las propiedades deseables en un LMP son (Conradi y Jaccheri, 1999):

- *Formalidad*: Tal como ya se ha dicho, el nivel adecuado de formalidad (es decir, la elección de un lenguaje formal, semi-formal o informal) depende de la fase del meta-proceso en que nos encontremos.
- *Expresividad*: Su máximo nivel se consigue cuando todos los elementos del MP pueden ser directamente representados por el LMP. Franch y Ribó (1999b) han propuesto algunas ideas para mejorar la expresividad en base al uso de lenguajes estándares modulares y flexibles.

- *Comprensibilidad* (facilidad de comprensión): Se ha comprobado que lo mejor es utilizar lenguajes gráficos basados en metáforas en el caso de usuarios no técnicos; y aprovechar la experiencia con lenguajes de programación en el caso de usuarios técnicos.
- *Abstracción*: El uso de modelos abstractos que se instancian en MP concretos permite manejar modelos a distintos niveles (ciclos de vida, modelos genéricos o detallados) y establecer correspondencias entre ellos.
- *Modularidad*: Estructurar un MP en sub-modelos interrelacionados ayudará a la reutilización de partes de un modelo.
- *Ejecutabilidad*: Definición de modelos operacionales que son ejecutables y fácilmente reificables.
- *Analizabilidad*: Definición de modelos descriptivos (por ejemplo, utilizando la lógica de predicados), que son fácilmente analizables.
- *Soporte de evolución*: La reflexión es muy importante para soportar la evolución de MP.
- *Múltiples vistas*: Soportar la definición de vistas consistentes desde diferentes perspectivas del MP. Para ello es necesario disponer de mecanismos de integración de vistas.

Al diseñar LMP que satisfagan las propiedades anteriores suficientemente se presentan bastantes problemas. En (Sutton et al, 1995a) se realiza un repaso de todos ellos y se detallan los más importantes: las dependencias entre los requisitos del proceso y los requisitos del lenguaje; la representación de los requisitos de ejecución; la amplitud, nivel y profundidad semánticos del lenguaje; y las meta-capacidades del proceso (metadatos, reflexión y dinamismo).

Son múltiples los campos de la Informática y de otras áreas de conocimiento que han aportado técnicas y herramientas útiles en los LMP. Los más significativos han sido:

- *Lenguajes de programación*.
- *Sistemas basados en reglas* (sistemas de producción, programación lógica, etc.).
- *Grafos y gramáticas*.
- *Redes de Petri*.
- *Orientación a objetos* (por ejemplo, utilizando extensiones de UML).
- *Notaciones estáticas* (entidad-interrelación).
- *Notaciones de comportamiento* (diagramas de transición de estados).
- *Notaciones funcionales* (diagramas de flujo de datos).
- *Gestión de proyectos* (diagramas de barras, redes de actividades, etc.).
- *Lenguajes de bases de datos* (bases de datos activas).
- *Herramientas CASE* y mecanismos de integración entre ellas.
- *Flujos de trabajo* (WorkFlows).
- *Herramientas de trabajo en grupo* (Groupware).

3.1.6.1. Taxonomía.

En general, los LMP se pueden dividir en tres categorías:

- *Formales*: Tienen sintaxis y semántica formales y, por tanto, proveen soporte para verificación y análisis formales, simulación y ejecución. En el apartado 3.1.6.2 se presenta un ejemplo de este tipo.
- *Semiformales*: tienen notación formal (normalmente gráfica) pero no tienen semántica formal. Esto último hace que no sean ejecutables. En el apartado 3.1.6.3 se comentan las propuestas de utilizar UML en este sentido.
- *Informales*: sin sintaxis y semánticas formales (por ejemplo, el lenguaje natural). Al igual que los anteriores, tampoco son ejecutables.

Los LMP formales pueden utilizar diferentes paradigmas lingüísticos. Los tipos más importantes de paradigmas propuestos por los investigadores e ingenieros son los siguientes (Xie, 2001):

- *Basados en Reglas*: que ponen el acento en las restricciones sobre la ejecución de los procesos. Ejemplos de este tipo son LATIN (Cugola et al, 1995), SPELL/EPOS (Conradi et al, 1992), MARVEL (Kaiser, 1993), que está basado en sistemas de producción, Merlin (Peuschel y Schäfer, 1992), MVP-L (ver apartado 3.1.6.2), Oikos (Ambriola et al, 1990), GOLOG (Plexousakis, 1995) y PEACE (Arbaoui y Oquendo, 1994).
- *Basados en Estados*: que modelan la estructura de roles y artefactos y sus interacciones. Ejemplos de esta clase son SLANG (Bandinelli y Fuggetta, 1993), que está basado en redes de Petri de alto nivel, y FUNSOFT (Deiters y Gruñi, 1994).
- *Funcionales*: que definen un proceso como una jerarquía de funciones matemáticas que representan las relaciones entre entradas y salidas. Hay pocas propuestas de LMP de este tipo; quizás las más conocidas son HFSP (Katayama, 1989) y ALF (Hanus, 1991).
- *Procedurales*: que modelan el flujo de control entre las actividades directamente como relaciones del tipo “saltar a”. Entre los lenguajes de esta clase se encuentran APPL/A (Sutton et al, 1995b), basado en el lenguaje Ada, y su evolución JIL (Sutton y Osterweil, 1997).
- *Orientados a Objetos*: que capturan los elementos de proceso como objetos y expresan las relaciones entre dichos objetos. Ejemplos de LMP de este tipo son E3 (Jaccheri y Stålhane, 2001), PROGRES (Schürr, 1996), SOCCA (Engels et al, 1998), Adele (Belkhatir et al, 1993), y ESCAPE+ (Reimer et al, 1997).

Algunos autores han buscado incorporar interfaces de usuario amigables de carácter gráfico para evitar que los usuarios se tengan que enfrentar directamente con la aridez del LMP formal. Esta opción es la seguida en el caso del lenguaje gráfico APEL (Dami et al, 1998), que incluye unos formalismos gráficos y un traductor a la versión textual del lenguaje basada en Adele.

También existen autores que han propuesto utilizar directamente lenguajes matemáticos en vez de elaborar LMP específicos. Un buen ejemplo de esto son las álgebras de procesos, que son conjuntos de notaciones formales y reglas para describir las relaciones entre procesos software. CSP (*Communicating Sequential Processes*) propuesto por Hoare (1995) y CCS (*Calculus of Communicating Systems*) propuesto por Milner (1989) son las dos propuestas de

este tipo más conocidas. Wang y King (2000) proponen una versión de CSP adaptada para modelar PS y sistemas de PS (ver apartado 3.1.4.4). Esta propuesta incluye lo siguiente:

- a) Una definición algebraica de *evento* y *proceso genérico*.
- b) Un conjunto de *primitivas de proceso* definidas a partir de las dos definiciones anteriores: ‘System-Dispatch’, ‘Assignment’, ‘Get-System-Time’, ‘Time-Synchronization’, ‘Event-Synchronization’, ‘Read’, ‘Write’, ‘Input’, ‘Output’, y ‘Stop’.
- c) Una serie de *operadores relacionales* para representar los diferentes tipos de relaciones que pueden existir entre procesos: dos tipos de secuencialidad (‘Serial’ y ‘Pipeline’); tres tipos de bifurcación (‘Event-Driven-Choice’, ‘Deterministic-Choice’ y ‘Nondeterministic-Choice’); tres tipos de paralelismo (‘Synchronous-Parallel’, ‘Asynchronous-Parallel-Concurrency’ y ‘Asynchronous-Parallel-Interleave’); dos tipos de iteración (‘Repeat’ y ‘While-Do’); dos interrupciones (‘Interrupt’ e ‘Interrupt-Return’); y dos tipos de recursividad (‘Guarded’ y ‘Generic-Recursive’).

Por ejemplo, el caso de un proceso P cuya ejecución significa la ejecución por dos veces de dos subprocesos P1 y P2 en paralelo, se representa con la siguiente fórmula:

$$P \wedge = (P1 \Downarrow P2) ; (P1 \Downarrow P2)$$

Desde un punto de vista diferente a los anteriores, otros autores han propuesto reutilizar lenguajes “inventados” inicialmente para otros dominios de aplicación, fundamentalmente la especificación de sistemas, y la representación de procesos de producción genéricos. Como ejemplo del primer caso, Podnar et al (2000) han propuesto SDL (*Specification and Description Language*), que permite el modelado formal de sistemas soportando concurrencia y orientación a objetos. Este lenguaje puede ser de ayuda en el caso del modelado de procesos complejos ya que permite una especificación jerárquica y modular. Un ejemplo del segundo caso es PSL (*Process Specification Language*), propuesto como estándar por el NIST norteamericano para representar procesos de fabricación (Schlenoff et al, 2000).

Además de los ya citados, en la bibliografía existen otras muchas propuestas de LMP software diferentes. En (Conradi y Jaccheri, 1999) se puede consultar una revisión de algunos de los lenguajes formales y semi-formales más conocidos. Un estudio detallado de las diversas aproximaciones para el modelado de procesos puede encontrarse en (Rombach y Verlage, 1995) y (Xie, 2001).

Además de la clasificación según nivel de formalidad ya citada, los LMP se pueden clasificar por otros criterios diferentes:

- a) Según el elemento del proceso en el que se centran:
 1. *Producto*: EPOS, Adele.
 2. *Actividades*: MARVEL, MERLIN, SLANG.
 3. *Proyecto*: MS-Project.
 4. *Roles*: PWI (*ProcessWise Integrator*).
- b) Según la fase del meta-proceso a la que se orientan:
 5. Orientados a la elicitación, análisis, diseño, implementación, reificación, o evaluación.

- c) Según otros aspectos alternativos:
6. *Funcionales*: centrados en describir las actividades.
 7. *De Comportamiento*: centrados en cuando y cómo se realizan las actividades.
 8. *Organizacionales*: centrados en el “cuando” y “por quién”.
 9. *Informacionales*: centrados en los artefactos y procesos, y las asociaciones entre ellos.

3.1.6.2. Ejemplo de Lenguaje Formal: MVP-L.

El lenguaje “*Multi View Process Modeling Language*” (MVP-L) tiene un formalismo basado en reglas, aunque también cuenta con esquemas de notación gráfica. Fue desarrollado por las universidades de Maryland (USA) y Kaiserslautern (Alemania) para el “modelado descriptivo de grandes procesos del mundo real para comprender, analizar, guiar y mejorar los proyectos de desarrollo de software” (Bröckers et al, 1995). Este objetivo general se concreta en los siguientes objetivos parciales: construir MP descriptivos; instanciarlos en planes de proyecto prescriptivos; empaquetar dichos planes de proyecto para su reutilización; analizar los planes de proyectos; orientar en la reificación de los proyectos; y documentar el histórico de reificaciones. Los cuatro primeros están orientados a la planificación y los dos últimos a la ejecución.

Para soportar estas metas un LMP debe permitir la representación de diferentes clases de modelos, la representación de MP, su integración e instanciación en planes de proyecto, y la construcción de MP complejos desde otros más simples. Por esta razón, MVP-L cuenta con los siguientes constructores:

- **Modelos**: Son tipos para la descripción de diferentes elementos de proceso:
 - *Modelos de proceso*: describen procesos, flujos de información, y descomposición del trabajo.
 - *Modelos de producto*: atributos de los productos creados durante la ejecución.
 - *Modelos de recursos*: herramientas y personas.
 - Estos modelos pueden ser adaptados al contexto actual durante su instanciación.
- **Atributos**:
 - Son globales o relacionados con uno de los 3 modelos.
 - Sus valores corresponden a medidas de datos y estados.
- **Objeto Ejecutable**: corresponde al *plan del proyecto*.
- **Relaciones** entre objetos:
 - *Flujos de producto*: relaciones del tipo consume, produce, etc.
 - *Flujos de control*: son criterios de entrada/salida de procesos o invariantes.
 - *Descomposición/Agregación*: basada en refinamientos.
 - *Ocultamiento* de información: se diferencia entre el interfaz de un modelo y el cuerpo de un modelo.

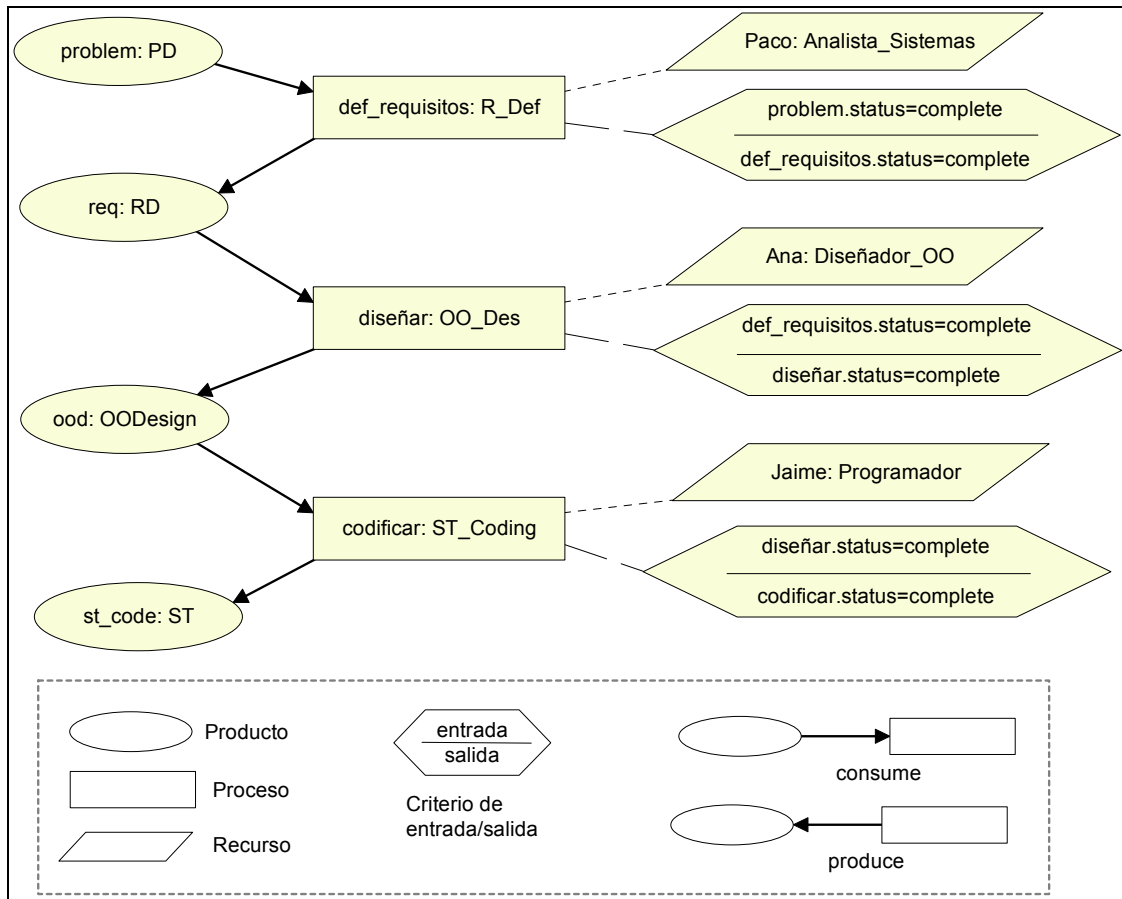


Figura 3-8. Diagrama de proceso en MVP-L.

En la Figura 3-8 se muestra como ejemplo el diagrama de un proceso sencillo (debajo se indica el significado de cada símbolo). Un posible plan de proyecto para ese mismo proceso, expresado con la notación utilizada en MVP-L, sería el siguiente:

```

project_plan
  imports
    product_model      PD, RD, OODesign, ST;
    process_model      R_Def, OO_Des, ST_Coding;
    resource_model     Analista_Sistemas, Diseñador_OO, Programador;
  objects
    problem: PD;
    def_requisitos: R_Def;
    req: RD;
    diseñar: OO_Des;
    ood: OODesign;
    codificar: ST_Coding;
    st_code: ST;
  object_relations
    def_requisitos(i1=>problem, o1=>req, r1=>Paco);
    diseñar(i1=>req, o1=>ood, r1=>Ana);
    codificar(i1=>ood, o1=>st_code, r1=>Jaime);
  end project_plan

```

3.1.6.3. Ejemplo de Lenguaje Semi-formal: UML.

Entre las diversas propuestas existentes de lenguajes orientados a objetos (OO) para modelar PS, UML ha sido una de las que ha recibido más atención. De hecho, al poco tiempo de publicarse la primera versión de UML ya surgieron propuestas de utilizarlo para este fin (Jäger et al, 1999). Las principales ventajas de utilizar UML para este fin estriban en su amplia utilización (es un estándar de facto), la existencia de un conjunto de diagramas para modelado conceptual y de comportamiento, y su soporte a las fases tempranas del modelado de procesos (análisis y diseño). El principal inconveniente es que UML (como la mayoría de lenguajes OO) no tiene una semántica bien definida y, por tanto, un MP representado con este lenguaje gráfico no es directamente interpretable en una computadora. Aunque existen otras propuestas de LMP gráficos (Dami et al, 1998), UML fue el elegido para representar los modelos y metamodelos de PS en este trabajo por las ventajas citadas. Como se detalla en el capítulo 4, el problema de su falta de rigor semántico para poder reificar los MP se ha abordado utilizando las capacidades que la tecnología de Flujos de Trabajo (*WorkFlows*) ofrece al respecto.

Franch y Ribó (1999a) proponen utilizar UML (Rumbaugh et al, 1999) para representar la parte estática de los procesos, es decir, los elementos que lo forman y sus relaciones estructurales. Estos autores consideran que la parte estática viene determinada por un modelo conceptual que define los elementos que forman el MP, mientras que la parte dinámica consiste en una descripción de la forma en que el modelo es reificado. Para ello han incorporado al lenguaje de modelado PROMENADE (PROcess-oriented Modelling and ENActment of software Developments) un metamodelo genérico que es una extensión del metamodelo general de UML y han propuesto un modelo de referencia (una instancia del metamodelo anterior) que constituye la base para construir cualquier MP específico. Este modelo de referencia es el núcleo común y extensible compartido por todos los procesos, incluyendo los elementos básicos que forman parte de cualquier MP descrito en PROMENADE. Estos elementos básicos son Tarea, Documento, Role, Herramienta, Recurso, Comunicación y Agente (Ribó y Franch, 2001); que claramente tienen una estrecha relación con los mostrados en la Figura 3-6. Estos elementos básicos se representan como clases abstractas en UML, es decir, son clases del metamodelo.

En un MP específico la clase “Documento” se especializa, utilizando el mecanismo de generalización, en los diferentes tipos de documentos o productos obtenidos. En el caso de las tareas, en PROMENADE se consideran dos aspectos relevantes para describir las tareas de un MP específico: la descripción de la tarea y la descomposición de la tarea. La primera, referida al comportamiento dinámico de las tareas, consiste en declarar las diferentes clases de relaciones de precedencia entre subtareas; mientras que la segunda sí que se refiere a la parte estructural (estática) del MP. La descomposición de tareas puede hacerse de dos formas diferentes:

- a) por *agregación*, es decir, indicando las tareas más simples que forman una tarea compuesta; y
- b) por *refinamiento*, es decir, especializando una tarea en varias subclases diferentes.

Mientras que la parte estática de un MP puede representarse utilizando diagramas de clase y objetos UML, según las indicaciones anteriores, no ocurre lo mismo con la parte dinámica (comportamiento). Por ejemplo, los autores anteriores opinan que los diagramas de actividad son los mejores en UML para representar la descripción de flujos de control, pero no tienen la expresividad requerida. Aunque algunas propuestas han ampliado UML con estereotipos (Rational, 1997), no resuelven el problema anterior porque las clases estereotipadas no tienen ni estructura ni comportamiento, y tampoco se permiten restricciones de integridad.

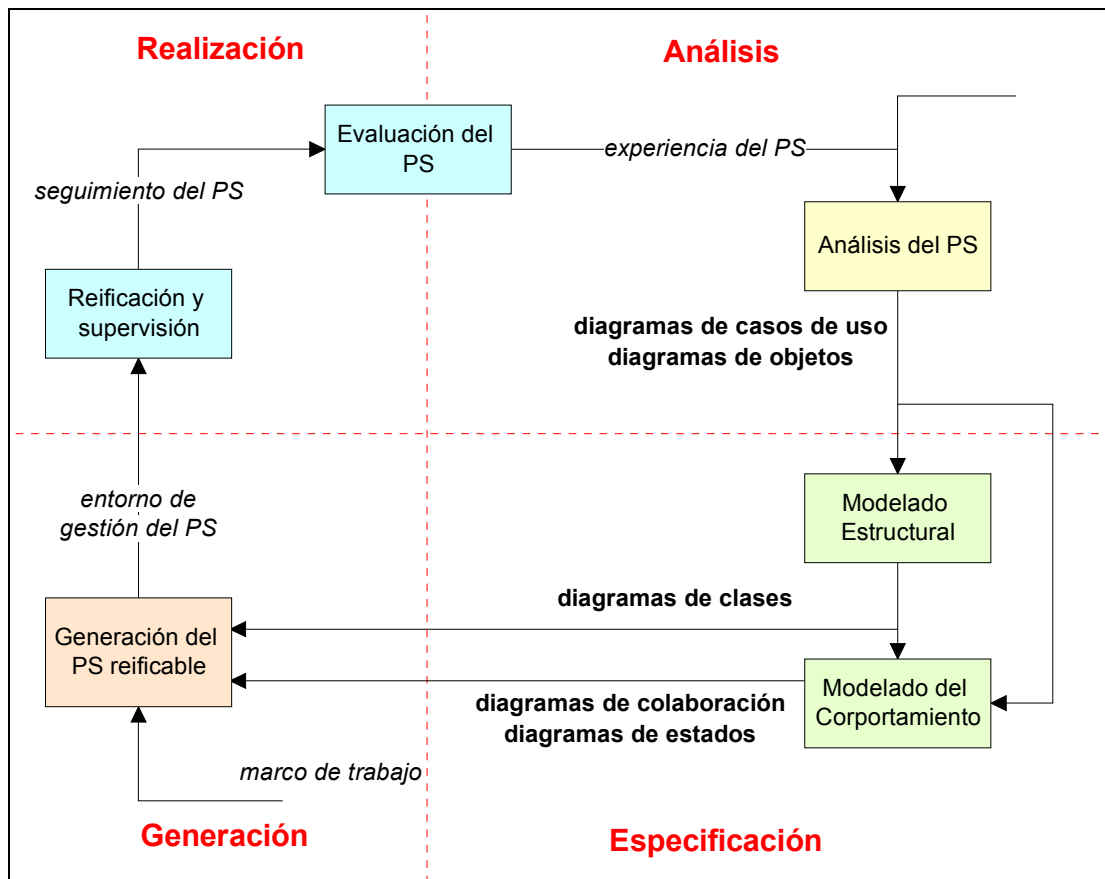


Figura 3-9. Metodología para Modelado de PS basado en UML.

En la Figura 3-9 se muestra una **metodología** iterativa e incremental propuesta por Jäger et al (1999) para el modelado de PS usando UML. De forma resumida, las cuatro etapas principales de esta propuesta son las siguientes:

- **Análisis:** El desarrollo de un nuevo MP comienza con la recopilación del conocimiento humano sobre el PS. Este conocimiento se representa mediante un diagrama de casos de uso para cada tipo de tarea del proceso. Las relaciones entre las tareas se representan mediante diagramas de objetos asociados a los diagramas de casos de uso.
- **Especificación:** Esta etapa se subdivide en modelado estructural y modelado del comportamiento. El primero consiste en indicar los elementos del MP, básicamente, los tipos de tareas existentes. Para ello se utilizan diagramas de clases donde cada clase representa un tipo de tarea. Después de indicar los elementos del MP, se puede pasar a especificar su comportamiento. Las transiciones de estado que pueden producirse se representan mediante diagramas de estados y las influencias mutuas entre tareas se representan mediante diagramas de colaboración.
- **Generación:** A partir del MP definido en UML en las dos etapas anteriores se genera un MP reificable, es decir, una versión en algún lenguaje interpretable y ejecutable en computadora. Esta transformación es posible gracias a la utilización de un metamodelo de PS genérico que provee la semántica necesaria, evitando el problema de la falta de formalidad de UML. Los autores de la propuesta han utilizado el lenguaje PROGRES para tal fin, que está basado en un sistema de reescritura de grafos (Schleicher, 1999).

- **Realización:** Puesto que es muy difícil capturar todos los aspectos de un proceso real de una sola vez, la metodología propone supervisar y registrar la reificación del proceso y evaluar los datos de seguimiento obtenidos. Con esta evaluación se procedería a realizar otra iteración empezando de nuevo por la etapa de análisis.

3.1.7. El Rol Humano en los Procesos Software.

La investigación en los aspectos sociales y humanos de los PS ha sido llevada a cabo principalmente por las comunidades de investigadores en Sistemas de Información, Ingeniería del Software, e Interacción Humano-Computadora (Arbaoui et al, 1999). En el ámbito de esta tesis, la que más interesa es la perspectiva aportada por la segunda, y más concretamente, la problemática originada al intentar aplicar un punto de vista humano (en vez del puramente tecnológico) en el diseño e implementación de EIS.

En esta línea, a continuación se presenta el marco conceptual de Downson (Downson y Fernström, 1994), que provee una base útil para representar las relaciones clave entre agentes humanos, EIS orientados a procesos (es decir, PSEE's), y modelos de proceso; y tener en cuenta estas relaciones al implementar el soporte a procesos en los EIS. Downson establece tres dominios diferentes en relación al PS (ver Figura 3-10):

- a) **Definición del Proceso**, que contiene las caracterizaciones de los procesos y fragmentos de procesos, expresadas en alguna notación, en términos de cómo dichos procesos podrían o deberían ser realizados.
- b) **Reificación del [Modelo del] Proceso**, que tiene que ver con lo que ocurre en un PSEE para soportar la realización del proceso.
- c) **Realización del Proceso**, que abarca las actividades reales o acciones dirigidas por agentes humanos (personal del proyecto: ingenieros software, gestores, etc.) y agentes no humanos (programas, herramientas del entorno, etc.).

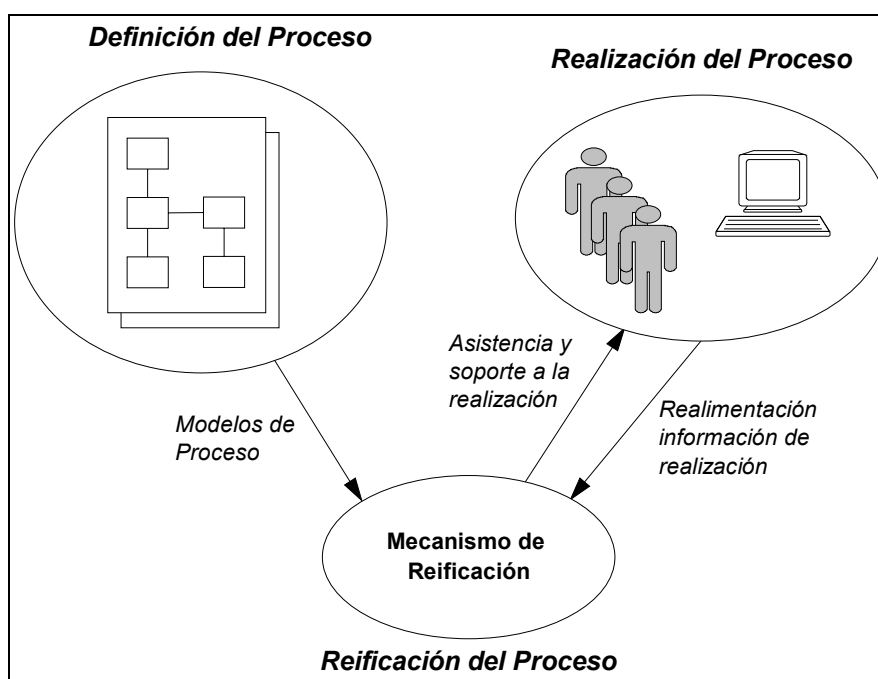


Figura 3-10. Dominios del Proceso Software según el marco conceptual de Downson.

Downson distingue cuatro roles diferentes desempeñados por agentes humanos en un PS (son roles abstractos que no tienen correspondencia unívoca con personas o grupos concretos):

- *Diseñador del Proceso*: actúa en el dominio de definición del proceso. Define MP genéricos y gestiona su evolución. Estos modelos definen una estrategia global para el desarrollo o mantenimiento del software independientemente del contexto específico de un proyecto (Benali y Derniame, 1992). Ejemplos de este rol son un diseñador de una metodología, un estratega técnico en una empresa de software, un consultor especializado en implantar programas de mejora de procesos software, etc.
- *Gestor del Proceso*: actúa en los dominios de definición y de reificación del proceso. Adapta los MP genéricos a un proyecto específico y los instancia con el fin de obtener un modelo reificable. En un caso real, este cometido puede ser realizado por un gestor de proyecto trabajando en conjunción con el director técnico de la empresa.
- *Agente del Proceso*: actúa en el dominio de realización del proceso. Puede ser un gestor de proyecto, un programador, un analista de sistemas, un auditor de calidad, un responsable de pruebas, etc.
- *Usuario Final*: actúa en el dominio de realización del proceso. Es el cliente, es decir, el usuario del producto que está siendo desarrollado o mantenido.

La realización del proceso pone en juego a los tres dominios citados, ya que todos los agentes mencionados, humanos o no, participan en un flujo dinámico de interacción (comunicando, coordinando, negociando, aconsejando, preguntando, etc.) dentro y fuera de los límites de los tres dominios. Estas interacciones pueden ser de dos tipos: interacción de usuario, e interacción interpersonal.

3.1.7.1. Interacción de Usuario.

Esta interacción se realiza entre los dominios de reificación y realización, es decir, entre los agentes de proceso y el MP reificado. Se refiere a actividades de carácter individual, como editar un módulo o leer un documento. Esta interacción es bidireccional:

- a) En el sentido “de *reificación a realización*”, se refiere a la influencia del mecanismo de reificación de procesos sobre los actores en orden a determinar la forma en que un proceso es realizado. En este asunto se plantea el problema de decidir quién ejerce el control durante la realización del proceso, el PSEE o un actor humano. Esta relación de poder pueda adoptar las formas siguientes (enumeradas de menor a mayor control por parte del PSEE):
 - *Guía Pasiva*: el mecanismo de reificación facilita a los actores, cuando lo piden, información para ayudarles a llevar a cabo el proceso.
 - *Guía Activa*: el mecanismo de reificación facilita ayuda sin que sea pedida por los agentes de proceso, es decir, el MP especifica cuándo y cómo debe ofrecerse dicha ayuda.
 - *Proceso Dirigido*: los agentes son obligados a realizar una parte del proceso de una determinada manera (por ejemplo, controlando su acceso a los datos y herramientas).

- *Proceso Automatizado*: alguna parte del proceso es realizada automáticamente bajo el control del mecanismo de reificación sin ninguna participación de los actores humanos.
- b) En el otro sentido, “de *realización a reificación*”, se refiere al enlace recíproco entre los agentes del proceso y el MP reificado en orden a proveer realimentación sobre el estado actual de la realización del proceso. Esta realimentación es crítica para asegurar que los “estados” de estos dos dominios estén alineados entre sí y sean tan consistentes como sea posible. Se pueden distinguir dos tipos de realimentación:
 - De primer orden, que se refiere simplemente a informes del progreso de acuerdo con el modelo prescrito; y
 - De segundo orden, que se refiere a los cambios locales que han sido hechos en el proceso en respuesta a exigencias prácticas (adaptación del proceso).

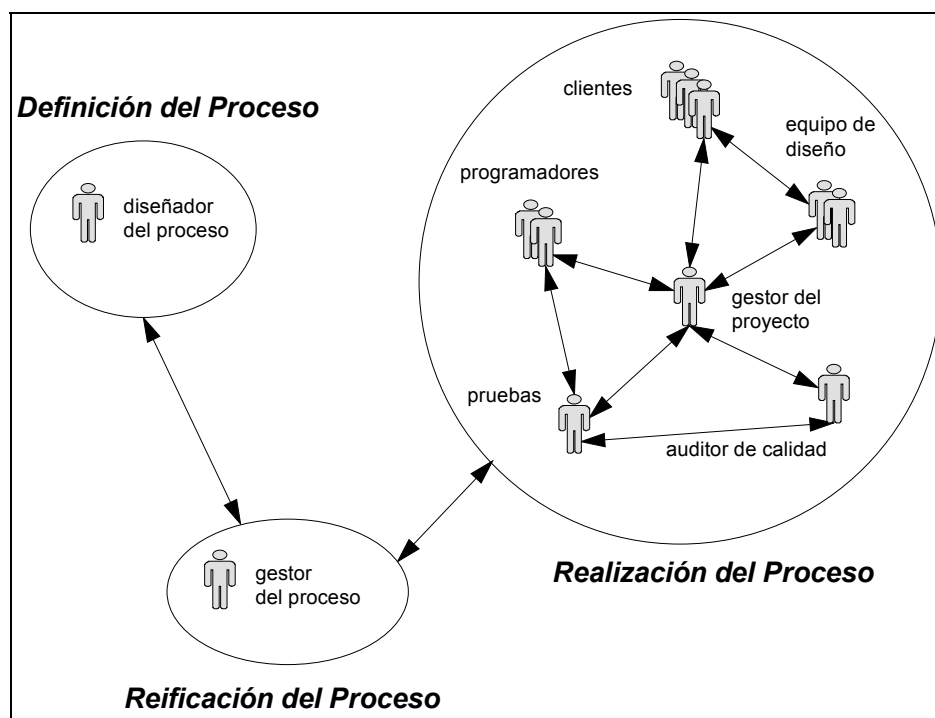


Figura 3-11. Patrones de interacción interpersonal en el proceso software.

3.1.7.2. Interacción Interpersonal.

Este tipo de interacción (ver Figura 3-11) se refiere a la interacción entre diferentes actores de proceso, por ejemplo, un analista comentando un documento de diseño con un programador, o una reunión para inspección de código (Gruñi, 1994). Se distinguen dos clases distintas de interacción interpersonal:

- *Formal*, si ocurre siguiendo unos protocolos bien definidos y formalizados. Un ejemplo de esta clase es una guía escrita que indica que si un programador tiene algún problema deberá informar al ingeniero de diseño.
- *Informal*, que ocurre cuando no está regulada por ninguna clase de protocolo de comunicación formal. Ejemplos de esta clase son una llamada telefónica, un correo electrónico, o una charla en la cafetería. La mayoría de interacciones en un proyecto software son de esta naturaleza y tienen lugar dentro del dominio de realización, ya que no

están mediatizadas o reguladas por el mecanismo de reificación y no se prevén en el diseño del proceso. Estas interacciones son cruciales para el éxito de un proyecto software ya que permiten que la información y la experiencia sean intercambiadas.

Las herramientas de soporte a las actividades interpersonales pueden tener un importante papel en el proceso software. De hecho, el campo del CSCW (*Computer Supported Collaborative Work*) ha sido la fuente de bastantes aportaciones útiles para dar soporte a las interacciones formales en un PS. En cambio, para las interacciones informales lo más importante es disponer de facilidades de comunicación en el dominio de realización (correo electrónico, boletines electrónicos, audio y/o videoconferencia, etc.).

3.2. Propuestas de Entornos.

A continuación se presentan algunas de las principales propuestas conceptuales y arquitecturales para EIS. En primer lugar nos referimos a la arquitectura PSEE para entornos orientados a procesos, la cual hemos incorporado al entorno MANTIS para la gestión del mantenimiento del software). También realizaremos un repaso de los modelos de referencia ECMA para marcos de trabajo EIS (es decir, para servicios de infraestructura básica de bajo nivel) y para servicios en entornos de soporte a proyectos (también llamados servicios de usuario o de alto nivel), y que, a juicio del autor, suponen la principal norma a tener en cuenta en cuanto a conceptos considerados y clasificación funcional de los servicios (en esto último complementados y mejorados por la propuesta del futuro estándar ISO 15940). Por último, presentamos la norma PCTE como ejemplo más conocido de interfaz de programación de servicios EIS. PCTE nos permite “aterrizar” concretando los conceptos y arquitecturas anteriores en definiciones precisas de los conceptos, utilizando un lenguaje de modelado, e indicando colecciones de operaciones, con un interfaz definido y conocido, para implementar los servicios EIS.

Aunque existen otras propuestas de arquitecturas y marcos de trabajo para EIS, los presentados en este documento permiten tener una visión general bastante adecuada de las distintas facetas implicadas en el área de los EIS. En Wallnau y Feiler (1991) se puede consultar la evolución desde las primeras propuestas de entornos integrados de soporte a proyectos (IPSE) hasta los modelos de referencia de servicios y marcos de trabajo ECMA, pasando por las coaliciones y federaciones de herramientas CASE. Otras propuestas, de interés para profundizar en este tema, son las siguientes:

- ATIS: “*A Tool Integration Standard*”, (Rock-Evans, 1993). ATIS es una especificación de un interfaz público para herramientas CASE, desarrollada por el “*CASE Integration Services*”, germen del comité técnico X3H6 de ANSI.
- CAIS-A: “*Common ‘Ada Programming Support Environment’ (APSE) Interface Set*”, (Munck et al, 1989). Esta es la propuesta de interfaz de programación de aplicaciones (basado en el lenguaje Ada) desarrollada por el ejército de Estados Unidos como una alternativa a la propuesta europea PCTE.
- PCIS: “*Portable Common Interface Set*”, (NATO, 1993). Esta propuesta surgió como una fusión de las primeras versiones de PCTE y de CASI-A, ya que se observó que entre ambas existían muchas similitudes.

- STAR: “*Software Technology for Adaptable, Reliable Systems*”, (USAF, 1996). Fue una iniciativa del ejercito americano buscando disponer facilidades de meta-programación.
- MetaCASE: No es una propuesta individual sino un área de trabajo centrada en la construcción de herramientas CASE adaptables a métodos de desarrollo de software diferentes (Koskinen, 1999).

Algunas experiencias de EIS concretos -orientados a procesos- se enumeran al final del apartado 3.2.1. Otros EIS que también aportan ideas interesantes son:

- MetaEdit: Es un entorno MetaCASE orientado a objetos que permite diferentes métodos en base a la utilización de un meta-modelo general (Kelly et al, 1996).
- Spearmint: Es un entorno orientado a una aplicación particular, el diseño y modelado de procesos software complejos, pero propone una arquitectura multicapa y multivista de aplicación general para EIS (Kempkens et al, 2000).
- IPSEN (*Integrated, Incremental, Interactive Project Support Environment*): Es un entorno de ingeniería del software que establece una arquitectura predefinida para integrar herramientas de programación como compiladores, depuradores, etc. (Nagl, 1996).

3.2.1. PSEE.

Ya se ha comentado que una característica principal de la tecnología de proceso software es el soporte a la automatización de procesos, en el sentido de ofrecer los medios adecuados para la definición, modificación, análisis y reificación de modelos de procesos. También se ha comentado el papel esencial que los PSEE (EIS orientados a procesos) juegan dentro de la TPS. Por esta razón, en los últimos años se ha desarrollado un importante esfuerzo investigador en el tema. Un importante resultado de este esfuerzo ha sido la publicación de un modelo de referencia y una propuesta arquitectural para entornos PSEE en general (Fugetta et al, 1999).

Un PSEE está controlado por un motor de procesos, es decir, un intérprete de un lenguaje de modelado de procesos. Esencialmente, el objetivo de este motor es el controlar el flujo de información entre los desarrolladores de acuerdo al modelo de procesos. El modelo es almacenado en un repositorio, junto con la definición del producto e información relevante sobre el estado del proceso. Además del repositorio, existen otro nivel de memoria importante formado por los espacios de trabajo, que son conjuntos de recursos informáticos que los desarrolladores utilizan cuando desempeñan un determinado rol en cierta actividad o tarea. Un PSEE también tiene que tener la capacidad para compartir datos con el exterior mediante canales de importación/exportación, que permitan el intercambio de productos y modelos en un formato de comunicación reconocible.

En la Figura 3-12 se resumen los componentes esenciales de esta propuesta. La línea discontinua desde el motor de procesos a la capa de comunicación indica que el motor de procesos controla el PSEE esencialmente controlando el flujo de información entre el repositorio y los espacios de trabajo, entre unos espacios de trabajo y otros, y entre los usuarios y sus espacios de trabajos.

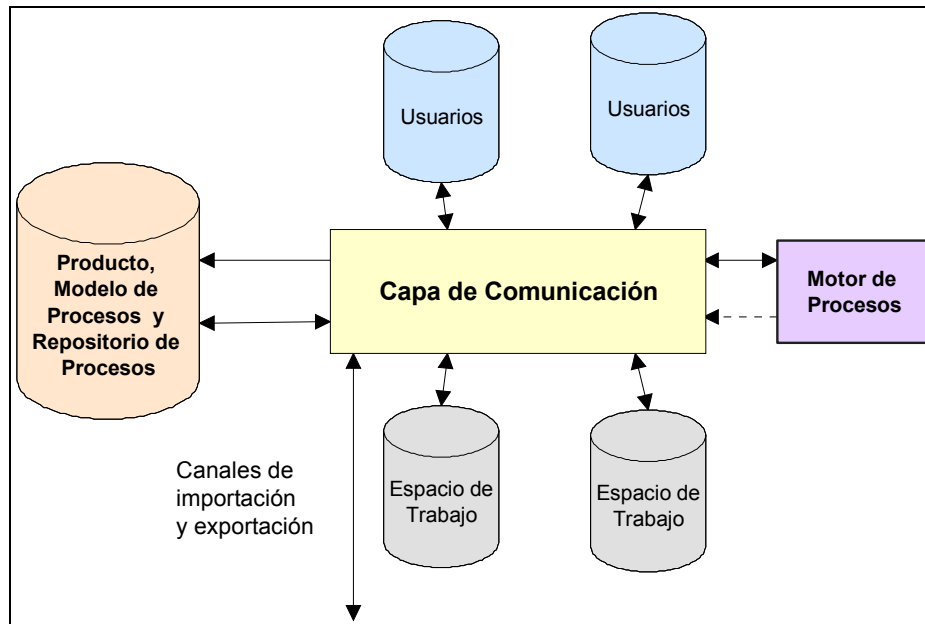


Figura 3-12. Arquitectura funcional de un PSEE.

Existe una similitud no casual entre esta arquitectura funcional y la de un computador en general. La razón es que un PSEE se puede considerar como una máquina abstracta para construcción de software, de forma que el modelo reificado juega el papel del programa almacenado, el repositorio es el equivalente a la memoria principal, los espacios de trabajo tendrían su correspondiente en los registros de máquina, y las herramientas tendrían el mismo rol que las operaciones primitivas. La diferencia fundamental viene dada por la diferente escala de tiempo con que ambos sistemas funcionan: fracciones de microsegundos para un computador frente a minutos, horas o incluso días, en un PSEE.

Para definir la arquitectura general de referencia para PSEE's han sido identificados los siguientes servicios básicos (Fugetta et al, 1999):

- Un *gestor del repositorio* PSEE para almacenar de forma persistente los objetos software y sus correspondientes relaciones, y poder acceder a ellos eficientemente.
- *Gestión de espacios de trabajo* personales, existiendo uno para cada usuario en cada uno de sus roles. Incluye todos los objetos software a los que tiene que acceder cada combinación usuario/rol.
- *Gestión de procesos*, cuya tarea es ejecutar un MP determinado y coordinar las actividades en curso de múltiples usuarios.
- *Gestión de diálogos*, para dar a los usuarios información sobre los procesos y permitirles llevar a cabo las actividades.
- *Gestión de comunicaciones*, que se necesita para intercambiar mensajes (notificaciones y peticiones) entre los diferentes componentes de un PSEE y las herramientas software integradas.

Como se muestra en la Figura 3-13, los cuatro primeros servicios básicos citados están organizados como una arquitectura de 4 capas de abstracción construidas una encima de otra. La mas inferior (interna) es la gestión del repositorio, que tiene encima a la gestión de espacios de

trabajo; ésta a la gestión de procesos, y por último, ésta a la gestión de diálogos. Existen varias alternativas sobre cómo estas capas de servicios pueden ser implementadas en componentes en un PSEE particular. Por ejemplo, en algunas propuestas de PSEE los espacios de trabajo son gestionados por el motor de procesos mientras que en otras son parte de un componente repositorio sofisticado. En la figura se muestra la opción recomendada, que asigna algunas funciones de la gestión de espacios de trabajo al motor de procesos y otras al componente encargado del repositorio. El quinto servicio básico, el gestor de comunicaciones, es necesario para intercambiar mensaje (notificaciones y peticiones) entre los diferentes componentes del PSEE entre sí y con las herramientas software integradas.

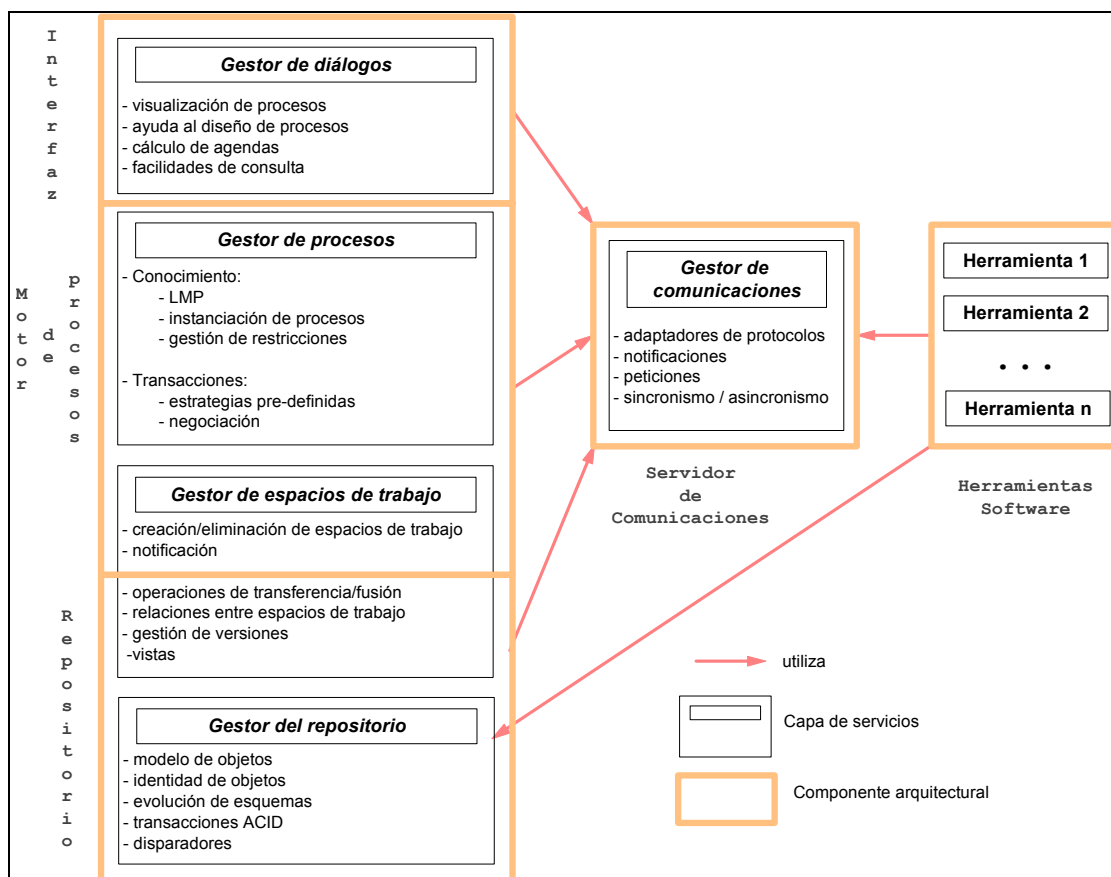


Figura 3-13. Arquitectura de referencia para PSEE.

La capa de gestión de diálogos encapsula la interfaz de usuario de un PSEE. Incluye soporte para visualización de procesos, asistencia en el diseño de procesos, cálculo de agendas, y facilidades de consulta. Dado que lo normal será que los usuarios de un PSEE interactúen con diferentes roles (desarrolladores, gestores de proyectos, ingenieros de procesos, etc.), se deberá disponer de un patrón de interacción diferente para cada rol.

La capa de gestión de procesos coordina las diferentes actividades de los múltiples actores involucrados en un proyecto software; y realiza los cálculos necesarios en el espacio de trabajo específico de cada usuario involucrado en un proyecto de desarrollo software, reflejando el estado actual del proyecto. Incluye ayudas para el conocimiento de los procesos: un lenguaje de modelado para representarlos, mecanismos de instanciación de procesos y gestión de restricciones (por ejemplo, para indicar mediante precondiciones los roles que pueden realizar

cada actividad). También incluye soporte para gestión de transacciones. Además, esta capa debe permitir expresar los estados actual y pasados de un proyecto.

En un PSEE puede haber un único motor de procesos central o varios distribuidos, ejecutando una descripción formal del PS, llamada “Programa del Proceso Software” (PPS). Un PPS debe poderse modificar durante la ejecución del proceso debido a que en los proyectos reales siempre hay circunstancias no previstas. Buscando una mejor gestión de los cambios, un PPS se puede dividir en tres partes:

- *Modelo de Cooperación*: que es la parte más estable, cambia sólo cuando cambia el paradigma de interfaz de usuario o se incorporan nuevas características como la gestión de la configuración. Provee un conjunto predefinido de reglas que constituyen la base para la ejecución de procesos, es decir, funciona como una máquina de inferencias para las otras dos partes.
- *Capa de Proceso*: es la vista normal del PS actual. La descripción en esta parte incluye, por ejemplo, tipos de documentos y herramientas, estados posibles, y transiciones de estado. Sus cambios ocurren durante la ejecución de un proyecto concreto porque ciertas partes de un proceso dependen de decisiones tomadas durante la ejecución real del proceso.
- *Capa de Proyecto*: Contiene la información necesaria para la instanciación del proceso: nombres, roles y responsabilidades de las personas que participan en un proyecto, y tipos y nombres de los documentos producidos. Sus cambios ocurren con bastante frecuencia debido a eventos imprevistos: rotura de una máquina, un accidente de una persona, detección de un error no conocido, etc.

La abstracción y el aislamiento son las dos motivaciones básicas que subyacen en la capa de gestión de espacios de trabajo, en el sentido de que los espacios de trabajo permiten a los usuarios concentrarse en sus tareas específicas dentro de un proyecto, abstrayéndose de la información irrelevante (para ellos) de otras partes del proyecto. Esto contribuye a preservar la integridad de los objetos del repositorio evitando su manipulación no intencionada o no autorizada. Pero, además de poder trabajar aislados cuando lo deseen, los participantes en un proyecto software también necesitan compartir objetos y sincronizar cambios en objetos compartidos (por más de un espacio de trabajo). Esta cooperación entre participantes múltiples debe ser coordinada por un mecanismo de transacciones avanzado (Godart, 1999). Esta capa incluye soporte para crear y borrar espacios de trabajo, notificaciones a los propietarios, operaciones de transferencia/fusión de espacios de trabajo, relaciones entre espacios de trabajo, gestión de versiones, y vistas de dichos espacios de trabajo.

El servicio de gestión del repositorio es responsable de mantener la consistencia y disponibilidad de la información que necesitan los otros componentes del PSEE. Por tanto, es una parte fundamental de un PSEE. En el repositorio se deben poder representar todos los tipos diferentes de datos (y metadatos) que habitualmente se generan durante un proyecto software. Incluye facilidades para el modelado de objetos, identidad de objetos, evolución de esquemas, transacciones ACID (*atomic, consistent, isolation, and durable*), y disparadores (*triggers*).

Aunque los SGBD's relacionales han demostrado de sobra sus capacidades de almacenamiento y procesamiento de datos, tienen dificultades para manejar las clases de datos generadas por un PSEE. Por esta razón, la mayoría de las propuestas de implementación de PSEE's tienden a utilizar SGBD's objeto-relacionales. Los requisitos exigibles a un gestor de repositorio para un PSEE son:

- *Composición y estructuración de datos*, pudiendo representar y manejar de forma eficiente todos los tipos diferentes de datos (y metadatos) que habitualmente se generan durante un proyecto software con orígenes heterogéneos. Los principales son:
 - *Datos de productos*: código fuente, datos de gestión de la configuración, documentación, ejecutables, juegos de pruebas, resultados de pruebas, simulaciones, ...
 - *Datos de procesos*: definición explícita de MP's, información del estado de la reificación de los procesos, datos para análisis y evolución de procesos, datos históricos, datos de gestión de los proyectos,...
 - *Datos organizacionales*: información sobre propietarios de componentes del proyecto, roles y responsabilidades, datos de gestión de los recursos, ...
- *Soporte multiusuario*, permitiendo acceso concurrente de múltiples usuarios.
- *Eficiencia*, cuando se recalculan los espacios de trabajo y estados de los proyectos ya que esto ocurre con mucha frecuencia. Por ejemplo, una instancia de proyecto cambia cuando se introduce un nuevo documento o un documento es borrado, cuando un documento es declarado como dependiente de algún otro documento, cuando un documento se vuelve incompleto debido a un cambio en algún otro documento del que depende, etc.
- *Persistencia e integridad*, ya que un PSEE puede necesitar pararse en algún momento y, por tanto, debe ser capaz de almacenar el estado del proyecto persistentemente para poder hacer después un “re-arranque”. También es necesario disponer de mecanismo que garanticen la integridad cuando se produzcan fallos en el sistema.
- *Distribución*, puesto que los sistemas multiusuario normalmente implican la distribución de las actividades del proyecto sobre varias estaciones de trabajo. Para los usuarios del PSEE debe ser transparente cómo está distribuida la información del repositorio.
- *Heterogeneidad*, ya que los diversos tipos de datos pueden tener formatos muy diferentes (ficheros ASCII planos, datos estructurados, tipos simples, objetos compuestos, objetos no estructurados, hipertexto, etc.).
- *Evolución*, porque todos los sistemas software representados en un PSEE evolucionan. Cuando alguna parte de un sistema es modificada, las demás partes dependientes de dicho sistema que se ven afectadas por el cambio se deben identificar, localizar y modificar para mantener al sistema como un todo consistente. Hay dos clases de cambios que debe gestionar un PSEE: en los datos de aplicación (por ejemplo, retrasar la fecha de finalización de una tarea) y en los metadatos o esquemas (por ejemplo, añadir una nueva tarea a un modelo de proceso).
- *Versionado y gestión de configuraciones*, ya que el repositorio tiene que soportar diferentes versiones de cada objeto y configuraciones del conjunto de objetos definidos en un proyecto.
- *Gestión de transacciones flexible*. El dominio de la ingeniería del software soportada por un PSEE está caracterizado por tareas complejas, de larga duración e interactivas. Para manejar y reducir esta complejidad, las tareas se suelen dividir en otras más simples, que pueden realizarse en paralelo, y que preservan la modularidad de diseño. Las especiales características de la interacción desarrollador-PSEE hace que no siempre sean las transacciones ACID las más adecuadas. Por ello, el gestor del repositorio debe poder utilizar también otros tipos de transacciones.

- *Facilidades de consulta “ad-hoc”*, porque la gestión de procesos necesita consultar la información del proyecto actual para extraer información de los estados de los documentos para que el motor de procesos pueda tomar decisiones. Una posible consulta sería la siguiente: ¿cuales son los módulos de los que es responsable Antonio, que están sin acabar, pero cuya especificación si está completa?.

La experiencia ha demostrado que los usuarios prefieren utilizar EIS que tengan una arquitectura abierta con diferentes niveles de integración. Esto se debe a que, normalmente, es necesario manejar distintos tipos de proyectos, a que el entorno debe ser adaptable a las necesidades cambiantes de un proyecto y a los nuevos avances tecnológicos, y a que debe ser posible añadir nuevas herramientas. Para integrar los diferentes componentes de su arquitectura el modelo de referencia PSEE propone el gestor de comunicaciones; otro componente que incluye soporte para adaptadores de protocolos, notificaciones y peticiones síncronas o asíncronas. Por otro lado, como una de las principales tareas de un PSEE es la coordinación de actividades llevadas a cabo por múltiples usuarios, el gestor de comunicaciones tiene que controlar la invocación y terminación de las distintas herramientas (editores, compiladores, generadores de casos de prueba, etc.) usadas para llevar a cabo las actividades que forman parte de los procesos software.

En Kobialka (1998) se puede consultar un resumen de los principales proyectos de desarrollo de PSEE's concretos. Algunos desarrollos de EIS adaptados ⁶ a la propuesta PSEE son los siguientes:

- *Process Weaver* (Fernström, 1993), utiliza en un LMP interno (oculto a los usuarios) basado en redes de Petri, que está centrado en las actividades. Esta orientado al desarrollo de software por grupos de personas.
- *Merlin* (Junkermann et al, 1994), permite el modelado de PS en tres niveles: un nivel para definir el estado del proyecto mediante predicados; un nivel de proceso con información independiente del proyecto (también usando predicados); y un nivel para gobernar la reficiación del proceso consistente en un conjunto de reglas tipo PROLOG. Una versión con arquitectura PSEE distribuida puede consultarse en (Fugetta et al, 1999).
- *Marvel* (Kaiser, 1993), incluye un LMP basado en reglas lógicas, cuyos tres constructores básicos son clases, reglas y herramientas.
- *SPADE* (Bandinelli et al, 1994), soporta análisis, diseño y reificación de PS. Los MP's se especifican usando el lenguaje SLANG, que utiliza un formalismo basado en redes de Petri de alto nivel. Permite modularización, instanciación y reflexión.
- *Serendipity* (Grundy y Hosking, 1998), provee modelado visual de procesos y lenguajes de manejo de eventos (ambos de alto nivel), y capacidades para trabajo colaborativo (CSCW), posibilitando la integración de herramientas software y la coordinación del trabajo automatizado con ellas.

⁶ Aunque todos estos entornos se originaron a principios de los años 90, cumplen del todo o en gran parte el modelo de referencia PSEE y han sido adaptados posteriormente a tal efecto. Algunos de ellos incluyen un LMP con igual o distinto nombre que el entorno.

3.2.2. ECMA.

Es interesante comparar la propuesta funcional de PSEE con la propuesta de ECMA (*European Computer Manufacturers Association*), conocida coloquialmente como modelo *tostadora* (ECMA, 1993). Mientras la segunda ofrece una perspectiva general de todos los servicios posibles de un EIS, clasificados en base a su funcionalidad, la primera se centra especialmente en los servicios de soporte a procesos. En resumen, la propuesta PSEE complementa y amplía la propuesta ECMA en servicios de proceso. Igualmente, la propuesta ECMA puede ser vista como una base para organizar la capa de comunicación de la Figura 3-12 y sus relaciones con el repositorio, los espacios de trabajo y las herramientas.

Al definir la arquitectura de un EIS es necesario diferenciar entre el conjunto de servicios que dan soporte al ciclo de vida de los proyectos (habitualmente implementados mediante herramientas software), y el conjunto de capacidades de infraestructura básicas necesarias para soportar procesos, objetos o interacción con los usuarios. En relación con los primeros, ya se ha comentado el estándar ISO 15940. Lo segundo se conoce como marco de trabajo (*Framework*) para EIS (Karrer y Penedo, 1990). Un marco de trabajo para EIS busca simplificar la construcción de herramientas (que implementan los servicios del EIS) proveyendo un conjunto de capacidades (que son necesarias casi siempre), ayudas para la integración, y soporte para constructores del alto nivel. Los componentes de un marco de trabajo EIS pueden ser utilizados por las herramientas del EIS o por otros componentes del marco de trabajo.

La propuesta ECMA para EIS consta de dos partes separadas, que presentamos a continuación. La primera propone un modelo de referencia para marcos de trabajo EIS y la segunda un modelo de referencia para entornos de soporte a proyectos.

3.2.2.1. Marcos de Trabajo para EIS.

Los primeros trabajos para crear el modelo de referencia para marcos de trabajo de EIS (*Reference Model for Frameworks of Software Engineering Environments*) comenzaron en Europa en 1989. A partir de 1991, esta propuesta (en adelante MT-ECMA) también fue adoptada en Estados Unidos por el NIST (*National Institute of Standards and Technology*). Este modelo es una base conceptual (y funcional) para describir y comparar EIS o componentes de EIS (incluidos componentes de marcos de trabajo). Sus principales objetivos son (ECMA, 1993):

- Servir para describir, comparar y contrastar marcos de trabajo EIS.
- Proveer una base para la identificación de estándares de interfaz EIS y sus relaciones.
- Orientar para la interoperabilidad e integración de herramientas.
- Ser útil para describir marcos de trabajo EIS diversos, buscando un adecuado equilibrio entre generalidad y especificidad.
- Ayudar a comprender las relaciones estáticas y dinámicas entre los componentes de un EIS.
- Ser independiente de las técnicas de implementación y de los métodos de construcción del software.
- Servir de base para la formación sobre EIS de los ingenieros software.

El modelo de referencia MT-ECMA consiste en un conjunto de servicios que corresponden a las capacidades funcionales de un marco de trabajo EIS. MT-ECMA describe un

marco de trabajo EIS en base a servicios. Estos servicios están clasificados en varios grupos para facilitar la comprensión. Por ejemplo, existen varios grupos que tienen que ver con las tres facetas de integración de herramientas ya comentadas: integración de la presentación (servicios de interfaz de usuario), integración del control (servicios de gestión de procesos y servicios de comunicación), e integración de datos (servicios de gestión de objetos). La lista de grupos y servicios para marcos de trabajo EIS propuestos en el modelo de referencia es la siguiente:

G1. Gestión de Objetos: servicios para la definición, almacenamiento, mantenimiento, gestión, y acceso a objetos y sus interrelaciones. Los servicios de este grupo son:

- *Metadatos*. Definición, control y mantenimiento de metadatos (por ejemplo, esquemas).
- *Almacenamiento de Datos y Persistencia*. Definición, control y mantenimiento de datos en base a los esquemas y tipos definidos previamente. Facilita persistencia de objetos, es decir, permite que los objetos tengan “vida” más allá del proceso que los creó.
- *Interrelaciones*. Capacidad para definir y mantener las relaciones entre objetos.
- *Nombres*. Soportar la asignación de nombres a los objetos y datos asociados, y mantener las relaciones entre los alias y los nombres. Evite que las personas tengan que utilizar los identificadores universales de objetos internos al sistema.
- *Distribución y Localización*. Capacidad para la gestión y acceso a objetos distribuidos. Permite que los datos -y posiblemente los servicios del marco de trabajo- puedan distribuirse entre una colección de procesadores y dispositivos del almacenamiento.
- *Transacción de Datos*. Capacidades para realizar una unidad de trabajo formada por una secuencia de operaciones atómicas.
- *Concurrencia*. Asegurar el acceso concurrente fiable (por usuarios o procesos) al SGO.
- *Soporte a Procesos del Sistema Operativo*. Proporcionar la habilidad para definir procesos del sistema operativo (es decir, objetos activos) y acceder a ellos utilizando los mismos mecanismos usados para los objetos.
- *Archivo*. Soportar la transferencia de información a los dispositivos de almacenamiento masivo y viceversa. Esto es necesario porque hay datos de un software que, una vez concluido su desarrollo, tienen que ser mantenidos durante muchos de años.
- *Copia de Seguridad*. Permitir la restauración del entorno de desarrollo a un estado consistente después de cualquier fallo en el sistema.
- *Derivación*. Definición y reificación de reglas de la derivación entre objetos, interrelaciones o valores (por ejemplo, atributos calculados, objetos derivados, u objetos heredados).
- *Replicación y Sincronización*. Permitir la replicación explícita de objetos en un entorno distribuido y la gestión de la consistencia de copias redundantes.
- *Control de Acceso y Seguridad*. Definición y aplicación de reglas para conceder o revocar derechos de acceso a los objetos del EIS a usuarios y herramientas.
- *Asociación de Funciones*. Permitir la asociación de funciones u operaciones a tipos de objetos, y la asociación de operaciones a instancias individuales de objetos. Estas capacidades están incluidas en el modelo de datos en el caso de los modelos OO.
- *Esquema Global*. Proporcionar un esquema global de los objetos y (posiblemente) de las descripciones de procesos en la base de datos que permite la integración de las

herramientas. Esto significa que todas las herramientas pueden usar el Esquema Global para describir y acceder a los datos que manipulan.

- *Versiones*. Capacidades para crear, acceder y relacionar versiones de objetos y configuraciones. Este servicio es necesario para poder incluir en un EIS la gestión de los cambios.
- *Objetos Compuestos*. Poder crear, gestionar, acceder, y eliminar objetos compuestos, es decir, objetos formados por otros objetos. Puede ser parte intrínseca del modelo de datos o ser un servicio separado.
- *Consulta*. Capacidad de recuperar conjuntos de objetos de acuerdo con unas propiedades y valores definidos.
- *Supervisión y Disparo de Estados*. Soportar la definición, especificación y reificación de estados de la base de datos y transformaciones de estado, y las acciones que deben realizarse para que estos estados ocurran o persistan.
- *Subconjuntos de Datos*. Permitir la definición, acceso y manipulación de un subconjunto del modelo de gestión de objetos (por ejemplo, tipos, interrelaciones, operaciones) o de instancias relacionadas (por ejemplo, objetos actuales).
- *Intercambio de Datos*. Soportar la traducción bi-direccional con repositorios de datos de otros EIS diferentes, es decir, poder intercambiar objetos con EIS basados en otros marcos de trabajo.

G2. Gestión de Procesos: servicios para la definición precisa y la realización asistida por computador (automática o semiautomática) de actividades de construcción y mantenimiento de software a lo largo de todo su ciclo de vida. Dichas actividades, además de las propiamente técnicas de desarrollo y mantenimiento, también incluyen la gestión, documentación, evaluación, estimación, cumplimiento de políticas, control de negocio, mantenimiento, etc. Los servicios de este grupo son:

- *Definición de Procesos*. Capacidad para crear, controlar y mantener definiciones de procesos (que pueden estar representadas formalmente). Estas definiciones corresponden a activos de proceso (*assets*) de una organización; que pueden ser definiciones de un proceso completo, un sub-proceso (elemento de proceso), un modelo de proceso, una arquitectura de proceso, o un diseño de proceso.
- *Reificación de Procesos*. Soportar la instanciación y ejecución de definiciones de proceso por agentes de proceso, que pueden ser humanos o máquinas. También proporciona capacidad para acceder, mantener, y controlar el estado persistente de los procesos.
- *Visibilidad de Procesos*. Proveer los medios para la definición y mantenimiento de información sobre la visibilidad y alcance asociada con los procesos reificados. Cuando varios procesos reificados cooperan para lograr un objetivo de más alto nivel; la magnitud de tal cooperación es parte de la definición de los procesos y puede proporcionarse mediante características de visibilidad integradas en las representaciones de la definición de un proceso particular. También se pueden manejar las interacciones inter-proceso proporcionando servicios independientes de visibilidad de datos comunes, eventos comunes, y propagación de información entre procesos.
- *Supervisión de Procesos*. Capacidad de observar la evolución en el estado de la reificación de procesos, detectar ocurrencias de eventos de proceso específicos, y reificar otros procesos para responder a estos eventos detectados.

- *Transacciones de Procesos.* Soportar la definición y reificación de transacciones de proceso, es decir, secuencias de pasos de proceso atómicos que deben ser completadas en su totalidad o deshechas para volver al estado previo su reificación.
- *Recursos de Procesos.* Asignación de agentes de proceso (herramientas, roles o usuarios individuales) para reificar procesos y elementos de proceso. Esta asignación se realiza sujeta a restricciones de tiempo, presupuesto, disponibilidad del personal, equipamiento, y tecnología para definición de procesos.

G3. Comunicaciones: servicios para la comunicación entre los servicios de un EIS. Los servicios de este grupo son:

- *Comunicación Inter-procesos.* Capacidad de comunicación entre procesos a nivel de sistema operativo mediante el mecanismo de RPC (*Remote Procedure Call*) del servicio de red.
- *Red.* Soportar la comunicación entre colecciones de procesadores heterogéneos.
- *Mensajería.* Permitir la comunicación entre procesos dentro de los elementos de un marco de trabajo EIS.
- *Notificación de Eventos.* Facilitar la notificación de mensajes basada en ciertas condiciones de disparo.

G4. Sistema Operativo: En cierto sentido, un EIS se puede considerar como una extensión del sistema operativo. Por ello, estos servicios describen una funcionalidad generalmente asignada al sistema operativo, pero cuya realización puede involucrar o no a dicho sistema operativo. Los servicios de este grupo son:

- *Gestión de Procesos del Sistema Operativo.* Proveer la habilidad de un sistema operativo para crear procesos y planificar su ejecución de forma independiente.
- *Entorno del Sistema Operativo.* Permitir el paso de información entre los procesos del sistema operativo.
- *Sincronización del Sistema Operativo.* Permitir la adecuada sincronización entre todos los procesos del sistema operativo en un EIS.
- *Entrada/Salida Generalizadas.* Soportar operaciones básicas para transferir datos entre procesos y unidades de entrada y salida asociadas a un EIS.
- *Almacenamiento de Ficheros.* Proporcionar las operaciones básicas para leer y escribir ficheros, almacenarlos y acceder a ellos en directorios.
- *Eventos Asíncronos.* Habilidad para crear y enviar señales entre procesos del sistema operativo.
- *Contador de Tiempo.* Habilidad para establecer y probar contadores de tiempo en los procesos del sistema operativo.
- *Gestión de Memoria.* Capacidad para gestionar la memoria principal del marco de trabajo.
- *Unidades Físicas.* Habilidad para gestionar los dispositivos físicos asociados al marco de trabajo.
- *Gestión de Recursos del Sistema Operativo.* Asignación de los recursos del sistema entre los diversos procesos del sistema operativo en un marco de trabajo.

G5. Interfaz de Usuario: servicios que proveen el principal medio para la interacción del usuario con el EIS. Estos servicios proporcionan el mayor canal de datos entre las herramientas y los usuarios. Los servicios de este grupo son:

- *Metadatos de Interfaz de Usuario.* Habilidad para definir ventanas, menús, iconos, etc.; y referenciarlos por su nombre. Este servicio permite definir esquemas utilizados por los otros servicios del grupo.
- *Sesiones.* Proporcionar la funcionalidad necesaria para comenzar y supervisar una sesión entre el usuario y el entorno.
- *Entrada de Textos.* Gestionar la entrada de datos de tipo carácter para el marco de trabajo.
- *Diálogos.* Proveer el interfaz entre los programas de aplicación y los dispositivos físicos de visualización. Este servicio permite pasar información de forma transparente entre los programas y los usuarios.
- *Gestión de la Visualización.* Permitir la interacción entre ventanas.
- *Presentación.* Capacidad para crear y gestionar el interfaz físico entre el usuario y el EIS. Esto incluye la pantalla y sensores de vista, sonido, tacto, etc., para interactuar con el EIS.
- *Seguridad del Interfaz de Usuario.* Facilita la mediación entre las vistas y acciones de usuario y las políticas de seguridad puestas en vigor en el marco de trabajo.
- *Nombres y Localización para el Interfaz de Usuario.* Facultar al marco de trabajo para gestionar entornos multi-usuario y multi-plataforma. Son necesarias varias sesiones para comunicar entre varias herramientas y varios dispositivos de visualización.
- *Internacionalización.* Proporcionar las capacidades necesarias para soportar convenciones locales para acceder a ciertos datos (por ejemplo, códigos de caracteres nacionales, formatos de fechas, etc.).
- *Ayuda al Usuario.* Proveer una realimentación consistente desde las diversas herramientas hacia el usuario para ayudarle e informarle de los errores.

G6. Implantación de Políticas: servicios que proveen soporte para la confidencialidad e integridad en un EIS, tanto para normas de seguridad obligatorias como discrecionales. Los servicios de este grupo son:

- *Información de Seguridad.* Permitir el establecimiento de información de seguridad para su uso dentro del EIS.
- *Identificación y Autenticación.* Habilidad para identificar a los usuarios y asociarles los privilegios de acceso correctos antes de que cualquier operación pueda ser llevado a cabo en su nombre.
- *Control de Acceso Obligatorio.* Capacidad para asignar privilegios de acceso por un responsable de seguridad para gobernar el acceso a la información contenida en un objeto.
- *Control de Acceso Discrecional.* Habilidad para tener privacidad personal.
- *Integridad Obligatoria.* Capacidad para proteger los objetos de modificaciones no autorizadas o no restringidas según determina el responsable de seguridad.

- *Integridad Discrecional*. Capacidad para proteger los objetos de modificaciones no autorizadas o no restringidas según determina un usuario.
- *Exportación/Importación Seguras*. Habilidad para exportar o importar objetos de forma segura.
- *Auditoría*. Habilidad para grabar información sobre las acciones que son tomadas que tienen relación con la seguridad.

G7. Administración del Marco de Trabajo: Un marco de trabajo EIS tiene que ser administrado cuidadosamente porque su configuración detallada puede estar cambiando continuamente para adaptarse a las necesidades cambiantes de una empresa de desarrollo o mantenimiento de software. Los servicios de este grupo son:

- *Registro de Herramientas*. Facilitar medios para incorporar nuevas herramientas en un EIS basado en el marco de trabajo, de forma tal que los diferentes componentes del marco de trabajo se coordinen eficazmente con la nueva herramienta.
- *Gestión de Recursos*. Ayudar a la gestión, modelado y control de los recursos físicos de un EIS.
- *Medición*. Soportar los medios para determinar la productividad, fiabilidad y efectividad de un marco de trabajo y del EIS construido sobre el.
- *Sub-Entornos*. Capacidad para dividir un EIS en sub-entornos separados y restringir el acceso de los usuarios a sólo un subconjunto de los recursos del EIS que están disponibles.
- *Gestión de Auto-configuraciones*. Permitir la existencia de muchas configuraciones residentes simultáneas de una misma implementación del marco de trabajo.
- *Gestión de Licencias*. Soportar la implantación de requisitos de licencias sobre componentes del EIS. Esto es especialmente necesario en el caso del software usado en red, ya que es habitual que la licencia de uso sólo sea para un subconjunto de los usuarios del EIS.

3.2.2.1.1. Dimensiones de Servicios.

El modelo de referencia MT-ECMA propone 8 dimensiones diferentes para describir servicios EIS desde puntos de vista distintos (ECMA, 1993):

1. Conceptual: Describe la semántica (funcionalidad) del servicio sin referirse a cómo es implementado o las formas de hacerlo disponible a los otros servicios.
2. Operaciones: Indica el conjunto de operaciones que implementan la funcionalidad descrita en la dimensión conceptual.
3. Reglas: Se refiere al conjunto de reglas que restringen los estados que los objetos pueden alcanzar y los cambios de estado que las operaciones pueden producir. Ejemplos de reglas son las pre-condiciones, post-condiciones, o restricciones en el uso de objetos u operaciones.
4. Tipos: Describe los posibles tipos de objetos (o modelos de datos) usados por el servicio, información sobre dichos tipos (metadatos), y los objetos utilizados por el servicio (instancias de dichos tipos).

5. Externa: Discute la manera en que el servicio se hace disponible para ser utilizado (por otros servicios, por herramientas, o por los usuarios directamente).
6. Interna: Se refiere a los aspectos de implementación.
7. Servicios relacionados: Es una descripción de cómo puede interactuar con otros servicios. Es interesante separar las relaciones estáticas entre servicios de las dinámicas. De esta manera, además de poder describir qué servicios se pueden usar con qué otros, también se pueden indicar las secuencias de interacciones que pueden tener lugar.
8. Ejemplos: se proporcionan ejemplos de servicios en el modelo de la referencia.

3.2.2.2. Soporte a Proyectos.

El objetivo del modelo de referencia ECMA (1994) para entornos de soporte a proyectos (*Reference Model for Project Support Environments*) es describir entornos de soporte a proyectos (ESP) de ingeniería informática (no sólo de ingeniería del software). Estos proyectos pueden ser muy variados, con grados de automatización muy diferentes y con una naturaleza distinta (estudio, construcción, mantenimiento, ...). Pero todos ellos tienen un conjunto de características comunes, que no son aplicables únicamente a proyectos software, sino también a proyectos que involucren hardware y firmware, ingeniería de sistemas, etc.:

- Su ámbito es la exploración, ingeniería, desarrollo, o mejora de un sistema informático.
- Requieren alguna forma madura de gestión.
- Existe un soporte informático al proyecto.
- Existe un soporte informático para comunicación durante la ejecución del proyecto.
- Existen varias fases durante el ciclo de vida del proyecto, a menudo varias actividades de ingeniería.

Los proyectos requieren muchos tipos de soporte. Las funciones de los proyectos que pueden ser soportadas por ESP's pueden agruparse en cuatro categorías principales:

- *Ingeniería Técnica* (diseño del sistema, simulación, ...).
- *Gestión Técnica* (gestión de la reutilización, gestión de configuraciones, ...).
- *Gestión del Proyecto* (planificación de recursos, seguimiento del proyecto, ...).
- *Soporte* (edición, mantenimiento de las facilidades, ...).

El modelo de referencia para ESP es una descripción conceptual de la funcionalidad que puede facilitar un ESP (o un EIS en el caso del interés de este trabajo), complementando la propuesta para marcos de trabajo EIS. Mientras que la propuesta ya presentada se refiere a los servicios de infraestructura básica para soportar EIS, ésta se refiere a los servicios de interés directo para el usuario final del entorno (ingeniero, gestor o cualquier otro que participe en la ejecución del proyecto). Además de este punto de vista conceptual, un entorno puede ser analizado desde otro punto de vista "actual" o real. En este caso, en vez de hablar de servicios se habla de herramientas, que son los componentes del entorno que directamente dan soporte a los usuarios. Cuando el entorno es analizado desde el punto de vista de su soporte a las actividades humanas, se considera que el entorno provee un servicio a un usuario humano o que un usuario humano realiza alguna tarea con la ayuda del entorno. En resumen, los servicios son las

capacidades del entorno, las tareas utilizan dichas capacidades, y las herramientas son los componentes software ejecutables actuales (reales).

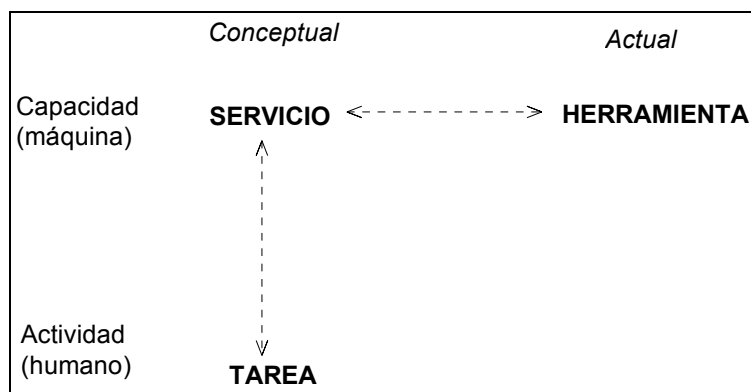


Figura 3-14. Relaciones entre Servicios, Herramientas y Tareas.

Las relaciones entre los conceptos de servicio, herramienta y tarea están resumidas en la Figura 3-14. En el plano horizontal, un servicio se “materializa” en una o varias herramientas software y éstas se conceptualizan en uno o varios servicios. A la vez, en el plano vertical, un servicio es una capacidad para realizar una o varias tareas y una tarea se puede realizar gracias a las capacidades ofrecidas por uno o varios servicios.

En los entornos reales no existe una diferencia nítida entre los grupos de servicios comentados. Por ejemplo, es frecuente encontrar herramientas que soporten servicios de usuario final junto con otros de infraestructura pertenecientes al marco de trabajo. En la Figura 3-15 se muestra esta dicotomía entre las descripciones conceptuales de servicios (donde no se produce duplicaciones de funcionalidad) y los componentes software de un entorno real, que solapan en sus funcionalidades con cierta frecuencia.

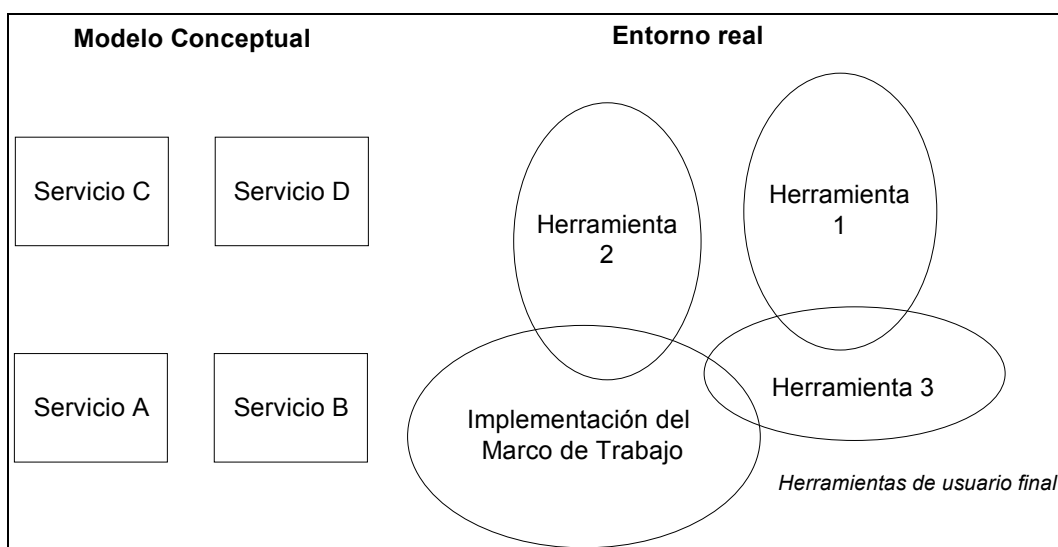


Figura 3-15. Grupos de servicios conceptuales y componentes software de un entorno real.

El catálogo de servicios de usuario final propuesto en el modelo de referencia para ESP es similar al modelo de servicios ISO 15940 ya presentado (apartado 3.1.3.3). Las principales diferencias estriban en que ISO se limita únicamente a EIS y por tanto, no incluye servicios de ingeniería de sistemas, y que ECMA establece una clasificación en tres niveles frente a los dos de ISO. La clasificación de servicios establecida por ECMA (sin detallar los servicios individuales) se muestra en la Tabla 3-3.

Servicios de Usuario Final	
Grupo	Subgrupos
Ingeniería Técnica	Ingeniería de Sistemas
	Ingeniería del Software
	Ingeniería de los Procesos del Ciclo de Vida (Gestión de Procesos)
Gestión Técnica	Gestión Técnica
Gestión del Proyecto	Gestión del Proyecto
Soporte	Soporte General
	Publicación
	Comunicación de Usuarios
	Administración del Entorno
Marco de Trabajo	Los establecidos en el modelo de referencia para marcos de trabajo EIS

Tabla 3-3. Clasificación de los servicios de usuario final en un Entorno de Soporte a Proyectos.

3.2.3. PCTE.

PCTE (*Portable Common Tool Environment*) es un interfaz a un conjunto de facilidades (equivalentes a los servicios en las propuestas anteriores) que sirven de base para la construcción de entornos de soporte a proyectos informáticos (incluidos los EIS). Estas facilidades están diseñadas de forma especial para ser una infraestructura útil a las herramientas que forman parte de dichos entornos y para soportar la portabilidad proveyendo acceso independiente de la máquina. En suma, se trata de un “interfaz de programación de aplicaciones” (API) especialmente diseñado para ayudar en la construcción de herramientas CASE y otras de soporte a la ingeniería informática en general.

PCTE tuvo su origen en un proyecto europeo ESPRIT el año 1986. Su éxito hizo que ECMA acogiera dicho proyecto a partir de 1988, surgiendo la primera versión del estándar en 1990. En la actualidad está disponible la versión cuarta (ECMA, 1997).

La arquitectura PCTE está descrita en dos dimensiones: estructural y funcional. La arquitectura estructural establece cómo una instalación PCTE está formada por un sistema de estaciones de trabajo intercomunicadas y cómo el software que provee los interfaces PCTE está estructurado. La arquitectura funcional ofrece una descripción de los componentes funcionales de PCTE y las facilidades que provee cada uno. En Long y Morris (1993) se puede profundizar en la historia de PCTE, sus relaciones y convergencias con otras propuestas, y las principales implementaciones existentes.

3.2.3.1. Arquitectura Estructural.

La arquitectura estructural propuesta para una instalación PCTE es un conjunto de estaciones de trabajo y de recursos asociados comunicados entre sí mediante una red, sin que exista una jerarquía entre dichas estaciones de trabajo (red *peer-to-peer*). Un usuario de una

estación de trabajo incluida en una instalación PCTE observa dicha instalación como si fuera una máquina conceptualmente única, aunque cada estación de trabajo puede actuar como una unidad autónoma. De esta manera, un usuario tiene acceso a todos los recursos de la instalación PCTE, aunque sujeto a los controles de acceso debidos.

La base de datos PCTE (repositorio) está dividida en volúmenes. Los volúmenes son situados de forma dinámica (montados) en estaciones de trabajo concretas y, una vez montados, están disponibles de forma global.

Un programador de herramientas para EIS no necesita tener en cuenta la arquitectura distribuida ya que los interfaces PCTE proveen todas las facilidades necesarias para configurar una instalación PCTE y controlar su distribución. Los interfaces PCTE aparecen a los desarrolladores de herramientas como disponibles dentro de una instalación PCTE independientemente de la situación física de la herramienta dentro de la instalación e independientemente de cualquier topología de red particular.

3.2.3.2. Arquitectura Funcional.

Los principales componentes funcionales de PCTE son los siguientes:

1. Sistema de gestión de objetos. Un aspecto de gran importancia para construir e integrar herramientas portables con PCTE es la disponibilidad de una base de objetos (repositorio de los datos usados por las herramientas en una instalación PCTE), y de un SGO que provee las funciones para acceder y manipular dicha base de objetos.
2. Base de objetos. El modelo de datos básico de PCTE está derivado de Entidad-Interrelación. Sus elementos básicos son objetos y enlaces. Los objetos son entidades que pueden tener contenidos (equivalente al concepto de fichero tradicional), atributos y enlaces (representaciones de asociaciones entre objetos).
3. Gestión de esquemas. PCTE provee facilidades para controlar la estructura, almacenar y nombrar los objetos de una manera uniforme. Hay definidas reglas de tipado (control y asignación de tipos) para objetos, enlaces y atributos. Para proveer vistas consistentes y autocontenidas de los datos en la base de objetos se definen SDS's (*Schema Definition Sets*). Además, cada proceso en ejecución ve la base de objetos a través de un "esquema de trabajo" (una composición de SDS's). También existe un "esquema global" compuesto por todos los SDS's de la base de objetos.
4. Auto-representación y esquemas predefinidos. Para muchas de las entidades habituales en una instalación PCTE existen SDS's predefinidos. Por ejemplo, los procesos son representados por objetos de tipo "proceso" que está definido en el SDS predefinido "system".
5. Contenido de objetos. Se proveen un conjunto de operaciones para acceder al contenido de algunos tipos de objetos (ficheros, dispositivos físicos, etc.). Estas operaciones proveen facilidades de entrada/salida tradicionales o especiales (por ejemplo, a la estructura interna de ficheros de auditoría).
6. Ejecución de procesos. Una ejecución de un programa (herramienta) es un proceso. Los procesos se representan mediante objetos en la base de objetos. Esto permite controlar, manipular y examinar sus características (procesos padre e hijos, entorno de ejecución, parámetros, fases por las que pasa, ...).
7. Supervisión de procesos. PCTE provee tres facilidades para la depuración y supervisión de procesos: medir los consumos de tiempo; observar y modificar la ejecución de un proceso hijo; y medir el uso de CPU.

8. Comunicación entre procesos. Los principales mecanismos para la comunicación entre procesos en PCTE son tres: los objetos, enlaces y atributos en la base de objetos; colas de mensajes; y tuberías (*pipes*).
9. Notificación. Este mecanismo permite que en un proceso se especifiquen eventos, correspondientes a operaciones sobre objetos, de los cuales debe ser notificado.
10. Control de concurrencia e integridad. La base de objetos está sometida al acceso concurrente de los usuarios. Además puede verse afectada por un fallo del sistema. Por estas razones, PCTE provee facilidades de bloqueo de objetos y asegura la consistencia e integridad de la base de objetos para transacciones atómicas y serializables.
11. Distribución. PCTE está basado en una comunidad de estaciones de trabajo, que pueden ser de diferentes clases, conectadas por medio de una red. Los objetos (incluidos los procesos) están distribuidos a lo largo y ancho de una instalación PCTE. Un usuario puede olvidarse de la localización de los objetos en volúmenes y de las estaciones de trabajo que están ejecutando los procesos; o puede ejercer un control sobre la localización de los objetos y los procesos.
12. Replicación. Estas facilidades permiten que en cada estación de trabajo se disponga de una copia local de los contenidos, atributos y enlaces de un objeto.
13. Seguridad. Una instalación PCTE tiene que soportar a muchos usuarios con diferentes roles dentro de diversos proyectos. Para ello son necesarias facilidades de seguridad que provean soporte para la definición de los privilegios de acceso de usuarios y herramientas.
14. Contabilidad. Estas facilidades permiten el registro automático de los consumos de determinados recursos por parte de los usuarios y herramientas.
15. Limites de implementación. Se ofrecen unos valores mínimos de ciertos parámetros de implementación, que si se cumplen se garantiza que la herramienta será portable a cualquier instalación PCTE.
16. Soporte a objetos de grano fino. Se introduce el concepto de “*cluster*” para mejorar el rendimiento. Un cluster es un objeto que representa a un conjunto de objetos de grano fino que comparten los mismos valores para ciertas propiedades.
17. Soporte de la orientación a objetos. Una importante característica de PCTE es su habilidad para definir cualquier modelo de datos de usuario y para utilizar una aproximación auto-referencial para definir sus metadatos. Las facilidades de orientación a objetos siguen una aproximación similar y describen todo como una extensión de la metabase o de la base de objetos.

Para cada uno de los componentes anteriores, se definen una serie de conceptos relacionados que se representan utilizando un lenguaje de definición de datos propio de PCTE; y una lista de las operaciones incluidas, indicando el modo de invocar cada una, sus parámetros y mensajes de error. A título de ejemplo, se muestran a continuación la definición del concepto de proceso (que utiliza el SDS predefinido “system”):

Initial_status = RUNNING | SUSPENDED | STOPPED

sds system:

inheritable: boolean := true;

```

referenced_object: (navigate) designation link (reference_name: string) to object with
attribute
    inheritable;
end referenced_object;

open_object: (navigate) designation link (open_object_key: natural) to file, pipe, device with
attribute
    opening_mode: (read) enumeration (READ_WRITE, READ_ONLY, WRITE_ONLY,
        APPEND_ONLY) := READ_ONLY;
    non_blocking_io: (read) boolean;
    inheritable;
end open_object;

is_listener: (navigate) non_duplicated designation link (number) to
    message_queue with attribute
    message_types: (read) string;
end is_listener;

process_waiting_for: (navigate) designation link (number) to object with
attribute
    waiting_type: (read) enumeration (WAITING_FOR_LOCK, WAITING_FOR_TERMINATION,
        WAITING_FOR_WRITE, WAITING_FOR_READ) := WAITING_FOR_LOCK;
    locked_link_name;
end process_waiting_for;

process: child type of object with
attribute
    process_status: (read) non_duplicated enumeration (UNKNOWN, READY, RUNNING,
        STOPPED, SUSPENDED, TERMINATED) := UNKNOWN;
    process_creation_time: (read) time;
    process_start_time: (read) time;
    process_termination_time: (read) time;
    process_user_defined_result: string;
    process_termination_status: (read) integer;
    process_priority: (read) natural;
    process_file_size_limit: (read) natural;
    process_string_arguments: (read) string;
    process_environment: (read) string;
    process_time_out: (read) natural;
    acknowledged_termination: (read) boolean;
    deletion_upon_termination: (read) boolean := true;
    time_left_until_alarm: (read) non_duplicated natural;
    character_encoding: (read) non_duplicated natural;
link
    process_object_argument: designation link (number) to object;
    executed_on: (navigate) designation link to execution_site;
    referenced_object;
    open_object;
    reserved_message_queue: (navigate) designation link (number) to message_queue;
    is_listener;
    default_interpreter: designation link to static_context;
    actual_interpreter: (navigate) designation link to static_context;
    process_waiting_for;
    parent_process: (navigate, delete) implicit link to process reverse child_process;
    started_in_activity: (navigate) reference link to activity reverse process_started_in;
component
    child_process: (navigate, delete) composition link (number) to process reverse
        parent_process;
    started_activity: (navigate) composition link (number) to activity reverse started_by;
end process;

end system;

sds metasds:

```

```

import object type system-process;

extend object type process with
link
    sds_in_working_schema: (navigate) designation link (number) to sds;
end process;

end metasds;

```

y la colección de operaciones que deben ser provistas por el componente de ejecución de procesos (el nombre de las operaciones y de los parámetros de entrada o salida permiten hacerse una idea aproximada de la funcionalidad aportada):

```

PROCESS_CREATE (static_context, process_type, parent, site, implicit_deletion, acces_mask)
    new_process
PROCESS_CREATE_AND_START (static_context, arguments, environment, site,
    implicit_deletion, acces_mask) new_process
PROCESS_GET_WORKING_SCHEMA (process) sds_sequence
PROCESS_INTERRUPT_OPERATION (process)
PROCESS_RESUME (process)
PROCESS_SET_ALARM (duration)
PROCESS_SET_FILE_SIZE_LIMIT (process, file_size_limit)
PROCESS_SET_OPERATION_TIME_OUT (duration)
PROCESS_SET_PRIORITY (process, priority)
PROCESS_SET_REFERENCED_OBJECT (process, reference_name, object)
PROCESS_SET_TERMINATION_STATUS (termination_status)
PROCESS_SET_WORKING_SCHEMA (process, sds_sequence)
PROCESS_START (process, arguments, environment, site, initial_status)
PROCESS_SUSPEND (process, alarm)
PROCESS_TERMINATE (process, termination_status)
PROCESS_UNSET_REFERENCED_OBJECT (process, reference_name)
PROCESS_WAIT_FOR_ANY_CHILD () termination_status, child
PROCESS_WAIT_FOR_CHILD (child) termination_status

```

3.3. Modelos de Proceso para Mantenimiento.

Entre las diversas propuestas o normas, publicadas a nivel internacional, útiles para modelar el proceso de mantenimiento del software (PMS), nos vamos a centrar en los estándares ISO correspondientes. Para su selección hemos tenido en cuenta que, desde una perspectiva funcional, se distinguen siete categorías diferentes de estándares para ingeniería del software (ISO/IEC, 2000a): proceso, producto, herramientas, tecnologías, recursos (humanos y materiales), y datos (de requisitos, de producto o de ingeniería). En la Figura 3-16 se muestra esta clasificación de forma esquemática. Una visión muy completa de los diferentes estándares de ingeniería del software propuestos por las principales organizaciones mundiales (IEEE, ISO/IEC, ESA, EIA, AIAA, PMI, etc.) puede consultarse en Moore (1998).

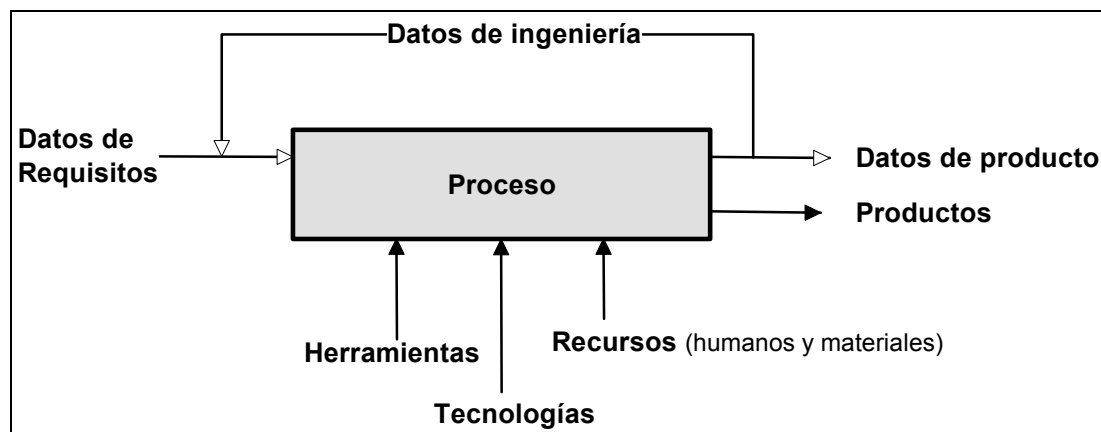


Figura 3-16. Clasificación de los estándares ISO para ingeniería del software.

Por razones evidentes, los estándares de más interés para este trabajo, que presentaremos a continuación, pertenecen a la categoría de proceso: ISO 12207 (1995) para los procesos que forman el ciclo de vida de un producto software, e ISO 14764 (1998e) para un modelo del proceso de mantenimiento. También incluimos la norma ISO 15504 para la evaluación de procesos porque, aunque está incluida en la categoría de tecnología, incluye un modelo de referencia para procesos software que completa el estándar 12207. En la Tabla 3-4 se presentan los diversos documentos oficiales de estos tres estándares.

Referencia	Fecha	Nombre
12207	1995	Information Technology - Software Life Cycle Processes
12207/AMD 1	2000	Amendment to ISO/IEC 12207:1995 - Information Technology - Software Life Cycle Processes
15271	1998	Information Technology - Guide for ISO/IEC 12207 (Software Life Cycle Processes)
15504-1	1998	Information Technology - Software Process Assessment -- Part 1: Concepts and Introductory Guide
15504-2	1998	Information Technology - Software Process Assessment -- Part 2: A Reference Model for Processes and Process Capability
15504-2/AMD 1	2001	Amendment To ISO/IEC TR15504-2 - Information Technology - Software Process Assessment - Reference Model Extensions For Acquirer Processes.
15504-9	1998	Information Technology - Software Process Assessment -- Part 9: Vocabulary
14764	1998	Software Engineering - Software Maintenance.

Tabla 3-4. Documentos sobre los estándares ISO 12207 y 15504 ⁷.

⁷ La norma ISO 15504 está formada por 9 documentos en total en su versión actual, aunque está en curso una revisión donde se reestructurará en sólo 5. En la tabla sólo se indican los documentos que tienen interés para el modelado de procesos software.

3.3.1. Ciclo de Vida: ISO 12207.

La norma ISO 12207 “establece un marco de referencia común para los procesos del ciclo de vida software, con una terminología bien definida, que puede ser referenciada por la industria software” (ISO/IEC, 1995). En este marco se definen los procesos, actividades (que forman cada proceso) y tareas (que forman cada actividad) presentes en la adquisición, suministro, desarrollo, operación y mantenimiento del software. Según esta norma, un proceso es un conjunto de actividades interrelacionadas que transforman entradas en salidas, definiendo quién está haciendo qué y cuándo, y cómo alcanzar un determinado objetivo. En la Tabla 3-4 se presentan los tres documentos oficiales referidos a esta norma: el propio estándar, una mejora posterior y una guía de utilización.

ISO 12207 establece una arquitectura de alto nivel del ciclo de vida de un producto software, desde su concepción hasta su retirada. La arquitectura está construida con un conjunto de procesos e interrelaciones entre ellos. La definición de dichos procesos está basada en dos principios elementales (ISO/IEC, 1998a):

- *Modularidad*: ya que todas las partes de un proceso están fuertemente relacionadas (cohesión fuerte) y el número de interfaces entre procesos se mantiene al mínimo (acoplamiento bajo).
- *Responsabilidad*: Cada proceso es responsabilidad de una parte ⁸, es decir, de un rol. Una organización puede realizar uno o varios procesos. Un proceso puede ser realizado por una o varias organizaciones (pero con una de ellas identificada como responsable). Este principio de responsabilidad facilita la adaptación a organizaciones y proyectos de cualquier tamaño y características.

Cada proceso se considera formado por una o varias actividades. A su vez, cada actividad está formada por una o varias tareas. Cada tarea se expresa en la forma de un requisito, una declaración, una recomendación o una acción permitida.

Esta norma clasifica los procesos que pueden ser realizados durante el ciclo de vida del software en cuatro categorías (ver Figura 3-17):

- **Procesos Principales**: Son procesos que dan servicio a las partes principales durante el ciclo de vida del software. Una parte principal es la que inicia o lleva a cabo el desarrollo, operación o mantenimiento de productos software. Estas partes principales son el adquirente, el proveedor, el desarrollador, el operador y el mantenedor de productos software.
- **Procesos de Soporte**: Apoyan a otro proceso, como parte esencial del mismo, con un propósito bien definido y contribuyen al éxito y calidad del proyecto software. Un proceso de soporte se emplea y ejecuta por otro proceso según sus necesidades.
- **Procesos Organizacionales**: Se emplean por una organización para establecer e implementar una infraestructura constituida por procesos y personal asociados al ciclo de vida, y para mejorar continuamente esta estructura y procesos.

⁸ Cuando una organización, ya sea completa o parcialmente, entra en un contrato, es una parte. Las organizaciones son entidades separadas, pero las partes pueden ser de la misma organización o de organizaciones distintas.

- **Proceso de Adaptación:** Este proceso especial define las actividades básicas necesarias para llevar a cabo adaptaciones de esta norma, es decir, adecuarla a las características especiales de cada proyecto u organización.

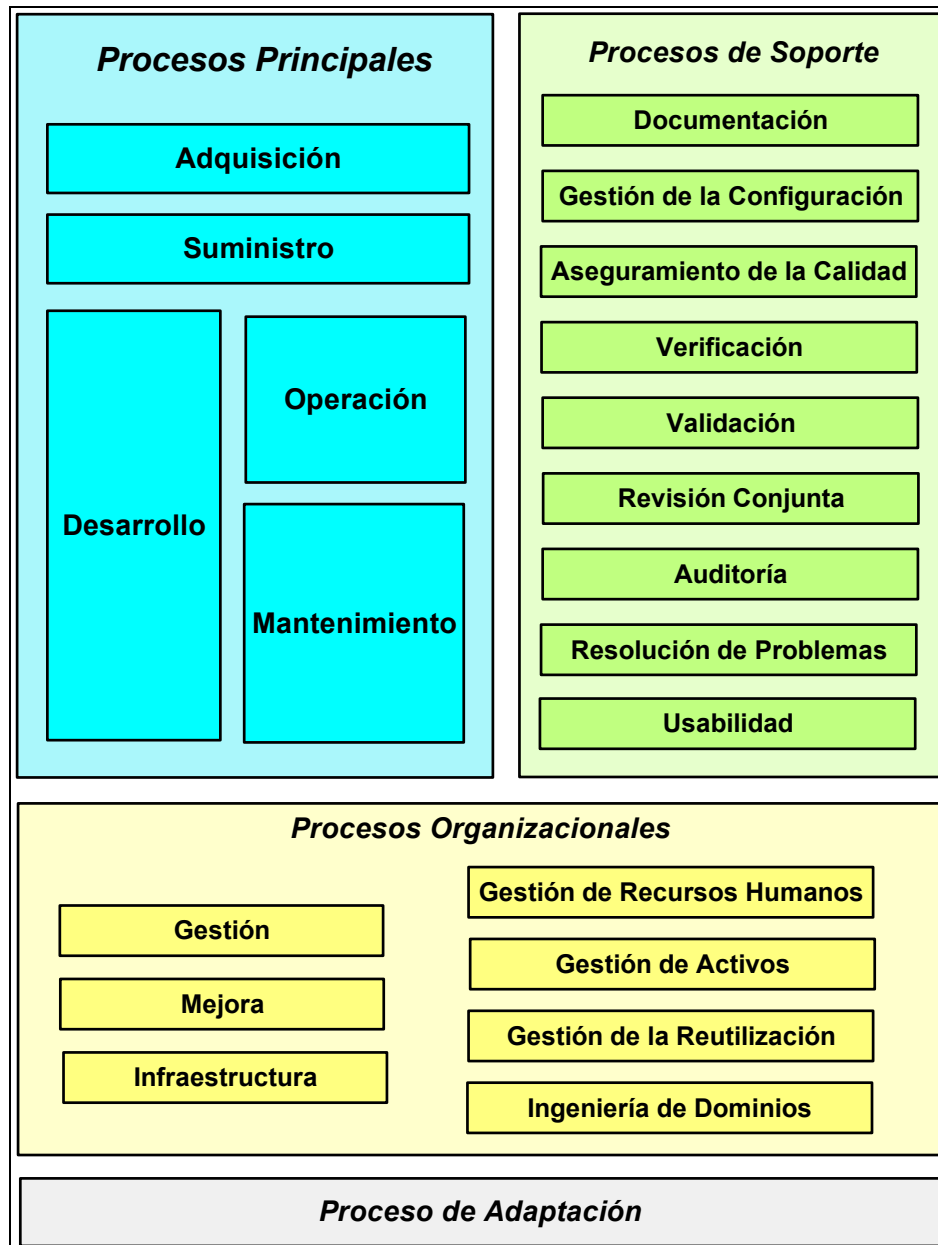


Figura 3-17. Procesos del ciclo de vida del software según ISO 12207.

3.3.1.1. Procesos Principales.

Como su nombre indica, son los procesos fundamentales en el ciclo de vida de un producto software. Los cinco procesos de esta clase son:

- Adquisición: Define las actividades del adquisidor (organización que adquiere un sistema, producto software o servicio software). Su objetivo es obtener el producto y/o servicio software que satisfaga una necesidad expresada por el cliente.
- Suministro: Define las actividades del suministrador (organización que proporciona el sistema, producto software o servicio software al adquisidor). Su objetivo es proveer un producto o servicio software que cumpla los requisitos acordados.
- Desarrollo: Define las actividades del desarrollador (organización que define y desarrolla el producto software). Su objetivo es transformar un conjunto de requisitos en un producto software funcional o un sistema basado en software, que satisfagan las necesidades establecidas por los clientes. Las actividades de este proceso son llevadas a cabo por uno de dos roles diferentes, el Desarrollador del Sistema o el Desarrollador del Software, aunque alguna actividad debe ser realizada por ambos (Evaluación del Producto).
- Operación: Define las actividades del operador (organización que proporciona el servicio de operar un sistema informático en su entorno real a los usuarios). Su objetivo es operar (explotar) el producto software en el entorno deseado y dar soporte a los clientes de dicho producto.
- Mantenimiento: Define las actividades del mantenedor (organización que proporciona el servicio de mantenimiento del producto software, esto es, la gestión de las modificaciones al producto software para mantenerlo actualizado y operativo). La finalidad de este proceso es ser sensible a las peticiones del cliente y gestionar las modificaciones, migraciones y retiradas de productos asociadas con ellas. El objetivo concreto es modificar y/o retirar sistemas o productos software preservando la integridad de las operaciones organizacionales.

3.3.1.2. Procesos de Soporte.

Los procesos de soporte se aplican en cualquier momento del ciclo de vida. Se pueden utilizar para dar ayuda a otro proceso de cualquier clase: principal, otro de soporte, organizativo o de adaptación. Los nueve procesos de esta categoría son los siguientes:

- Documentación: Define las actividades para el registro de la información producida por un proceso del ciclo de vida. Su propósito es elaborar y mantener registrada la información producida por un proceso.
- Gestión de la Configuración: Define las actividades de gestión de la configuración. Su objetivo es establecer y mantener la integridad de todos los productos de trabajo de un proceso o proyecto, y hacerlos disponibles a las partes interesadas.
- Aseguramiento de la Calidad: Define las actividades para asegurar, de una manera objetiva, que los productos de trabajo y los procesos son conformes a la especificación de requisitos y se ajustan a los planes establecidos. Son técnicas útiles en este proceso las revisiones conjuntas, auditorías, verificaciones y validaciones.
- Verificación: Define las actividades (para el adquisidor, suministrador u otra tercera parte independiente) para verificar los productos software hasta un nivel de detalle adecuado al proyecto software de que se trate. Su objetivo es confirmar que cada producto de trabajo y/o servicio software de un proceso o proyecto cumple adecuadamente los requisitos especificados.

- Validación: Define las actividades (para el adquisidor, suministrador u otra tercera parte independiente) para validar los productos software adecuados para un determinado proyecto. Su propósito es confirmar que los requisitos para un uso específico deseado de un producto de trabajo software son satisfechos.
- Revisión Conjunta: Define las actividades para evaluar el estado y productos de una actividad. Este proceso puede ser empleado por dos partes cualesquiera, donde una de las partes (la revisora) revisa a la otra parte (la revisada), de una manera conjunta. El objetivo de este proceso es mantener un conocimiento compartido con el cliente de los progresos realizados frente a los objetivos del contrato, así como de lo que debería hacerse para garantizar el desarrollo de un producto que satisfaga al cliente. Las revisiones conjuntas pueden ser a nivel técnico o de gestión del proyecto y ocurren a lo largo de toda la vida del proyecto.
- Auditoría: Define las actividades para determinar –de forma independiente- el cumplimiento por parte de los productos y procesos seleccionados de los requisitos, planes y contratos. Este proceso puede ser empleado por dos partes cualesquiera, donde una parte (la auditora) audita los productos software o actividades de otra parte (la auditada).
- Resolución de Problemas: Define las actividades para analizar y eliminar los problemas (incluyendo las no conformidades) que sean descubiertos durante la ejecución de otros procesos (desarrollo, operación, mantenimiento, etc.), cualesquiera que sea su naturaleza o causa. Su objetivo es asegurar que todos los problemas descubiertos son analizados y resueltos.
- Usabilidad: Incluye las actividades necesarias para asegurar que se están teniendo en cuenta los intereses y necesidades de los individuos y /o grupos que trabajarán con el sistema (usuarios). Este proceso garantiza la calidad de uso del software.

El proceso de Usabilidad es posterior al estándar oficial (ISO/IEC, 1995). Este proceso ha sido incorporado en la mejora del estándar (ISO/IEC, 2000b).

3.3.1.3. Procesos Organizacionales.

Los procesos de esta clase ayudan a establecer, implementar y mejorar la estructura y los procesos de una organización, contribuyendo a hacerla más eficiente. Habitualmente se llevan a cabo a nivel organizacional (corporativo), fuera del ámbito de proyectos y contratos específicos. Los siete procesos de esta clase son:

- Gestión: Define las actividades básicas de gestión (gestión de proyectos, gestión de calidad, gestión de riesgos, ...) durante un proceso del ciclo de vida. El objetivo general es organizar, supervisar y controlar la iniciación y realización de cualquier proceso o función para alcanzar sus objetivos en consonancia con los objetivos de negocio de la organización. Este proceso es establecido para asegurar la aplicación consistente de prácticas por parte de la organización en cada uno de los proyectos que lleva a cabo.
- Mejora: Define las actividades básicas que una organización (adquisidor, suministrador, desarrollador, operador, mantenedor o el gestor de otro proceso) lleva a cabo para establecer, evaluar, medir, controlar y mejorar un proceso del ciclo de vida del software.
- Gestión de Recursos Humanos: Define las actividades para conseguir personal formado adecuadamente. El objetivo es proveer a la organización y a los proyectos con individuos que posean habilidades y conocimientos para realizar sus roles de forma eficiente y para poder trabajar juntos como un grupo cohesionado.

- Infraestructura: Define las actividades básicas para establecer la infraestructura de un proceso del ciclo de vida. El propósito de este proceso es mantener una infraestructura estable y fiable necesaria para poder realizar cualquier otro proceso. Son infraestructuras para el ciclo de vida del software las siguientes: hardware, software, métodos, herramientas, técnicas, estándares, y otros medios útiles para desarrollar, operar o mantener software.
- Gestión de Activos: Define las actividades necesarias para gestionar la vida de activos (*assets*) reutilizables, desde su concepción hasta su retirada. Consiste en aplicar procedimientos administrativos y técnicos para su identificación, definición, certificación, clasificación, modificación, almacenamiento, recuperación y retirada.
- Gestión de la Reutilización: Define las actividades para planificar, establecer, gestionar, controlar y supervisar un programa de reutilización en la organización, y el aprovechamiento sistemático de las oportunidades de reutilización.
- Ingeniería de Dominios: Define las actividades para desarrollar y mantener modelos de dominios, arquitecturas de dominios y activos para los dominios.

La mejora del estándar (ISO/IEC, 2000b) ha cambiado el nombre del proceso de “Formación” a “Gestión de Recursos Humanos”. También ha añadido tres procesos nuevos: Gestión de Activos, Gestión de la Reutilización, e Ingeniería de Dominios.

3.3.1.4. Roles y Organizaciones participantes.

En la Figura 3-18 se representan los procesos del ciclo de vida del software y sus relaciones bajo diferentes puntos de vista de uso: contrato, gestión, operación, ingeniería y soporte. Cada punto de vista viene determinado por los intereses de un rol (parte) distinto. Desde la visión contractual, los roles adquisidor y suministrador negocian y se someten a un contrato empleando los procesos de Adquisición y Suministro, respectivamente. Bajo el punto de vista de gestión, una de las partes (adquisidor, suministrador, desarrollador, operador, mantenedor u otro) gestiona sus respectivos procesos. Bajo la visión de operación, el operador proporciona el servicio de operación del software para sus usuarios. Bajo el punto de vista de ingeniería, el desarrollador o mantenedor lleva a cabo sus respectivas tareas de ingeniería para producir o modificar los productos software. Por último, desde la visión de soporte, unos roles proporcionan servicios de apoyo a otros roles para completar tareas únicas y específicas. También se muestran (recuadro inferior) los procesos organizacionales, que se emplean por una organización, a nivel corporativo, para establecer e implementar la estructura subyacente compuesta por los procesos y el personal asociados al ciclo de vida y mejorarlos continuamente.

Aunque no se muestra en la figura, se puede considerar otro punto de vista de gestión de la calidad que engloba a los siguientes procesos de soporte: Aseguramiento de la Calidad, Verificación, Validación, Revisión Conjunta, Auditoría y Usabilidad. Todos estos procesos se emplean para gestionar la calidad a lo largo del ciclo de vida del software. Los procesos de Verificación, Validación, Revisión Conjunta y Auditoría se pueden emplear por diferentes roles separadamente (auditor, desarrollador, cliente, etc.) o como técnicas del proceso de Aseguramiento de la Calidad.

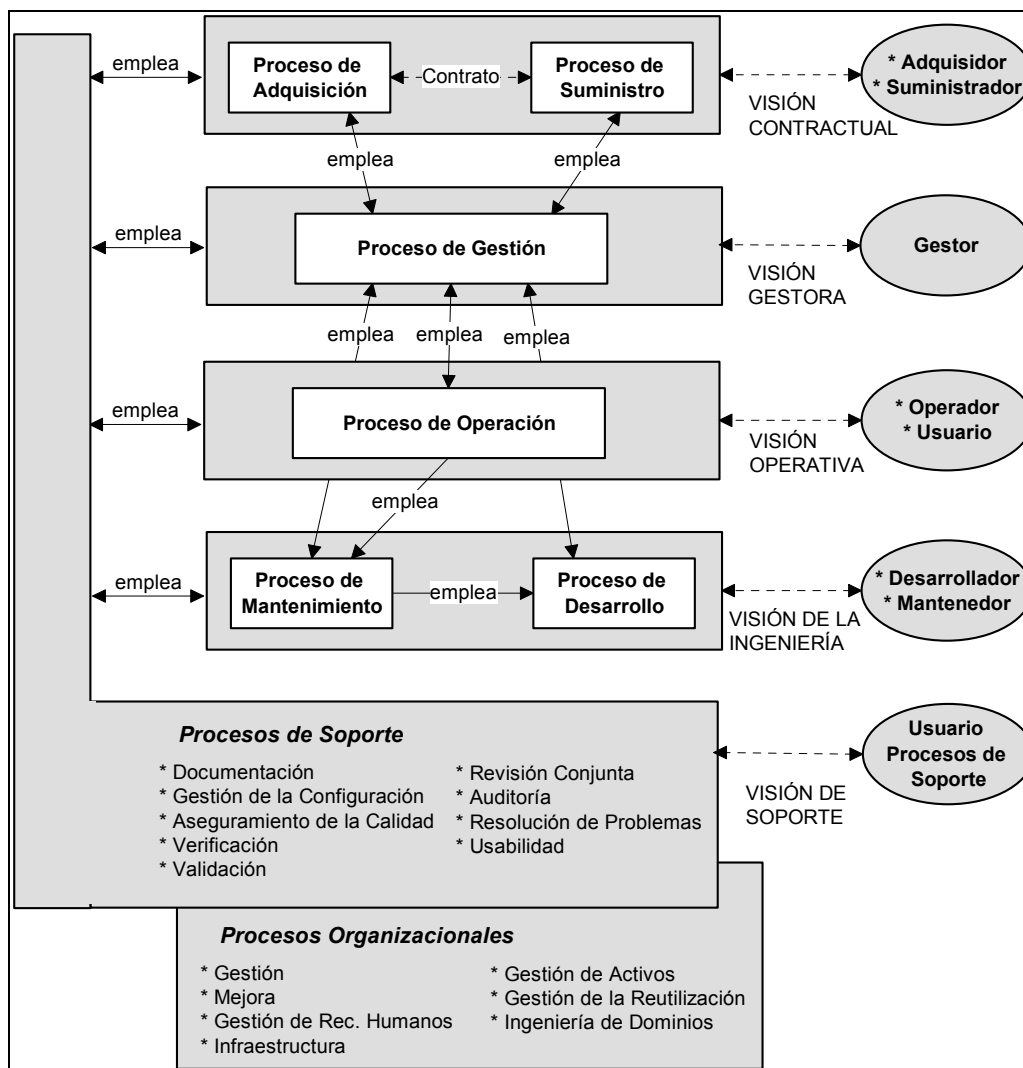


Figura 3-18. Roles y puntos de vista de los procesos del ciclo de vida.

Los procesos y organizaciones (o roles) están sólo relacionados funcionalmente, es decir, los puntos de vista anteriores no implican ninguna estructura para ninguna organización o rol. Una organización (o un rol) toma el nombre del proceso que lleva a cabo; por ejemplo, se le llama *adquisidor* cuando lleva a cabo el proceso de *Adquisición*. Una organización puede llevar a cabo uno o varios procesos, es decir, puede desempeñar uno o varios roles. Un proceso puede ser llevado a cabo por una o varias organizaciones, es decir, un mismo rol puede ser desempeñado por varias organizaciones. La única restricción a lo anterior es que una misma parte no debería llevar a cabo simultáneamente los procesos de *Adquisición* y *Suministro* del mismo producto o recurso.

En este modelo de ciclo de vida, las relaciones entre procesos son sólo estáticas. Las relaciones dinámicas más importantes de la vida real, entre procesos, entre roles y entre procesos y roles se establecen automáticamente cuando este modelo se aplica en proyectos software concretos.

3.3.2. Modelo de Referencia para Procesos: ISO 15504.

Para mejorar la calidad de un producto, una organización debe tener un método (probado, consistente y fiable) para evaluar el estado de sus procesos, y debe tener medios para usar los resultados como parte de un programa de mejora. Dentro de una organización, la evaluación de procesos incentiva tres aspectos importantes: la cultura de la mejora constante y el establecimiento de los mecanismos apropiados para apoyarla y mantenerla; la ingeniería de procesos para satisfacer los requisitos del negocio; y la optimización de recursos. Por tanto, la evaluación de procesos es importante para que las organizaciones maximicen su sensibilidad frente a los clientes y los requisitos de mercado, minimicen los costes del ciclo de vida de sus productos y, como resultado de ambos, mejore la satisfacción del usuario final.

El estándar ISO 15504 (ISO/IEC, 1998b, 1998c, 1998d) aborda la estimación de procesos software desde dos perspectivas diferentes, la mejora de procesos y la determinación de la capacidad de las organizaciones (ver Figura 3-19). Aunque está formado por 9 partes diferentes, las que son de interés en este trabajo son las partes 1 (introducción y conceptos), 2 (modelo de referencia para procesos y capacidad de procesos), y 9 (vocabulario). En la Tabla 3-4 se indican estos documentos junto con una mejora posterior del modelo de referencia de procesos software extendiendo los procesos propios del rol de adquisidor (ISO/IEC, 2001a).

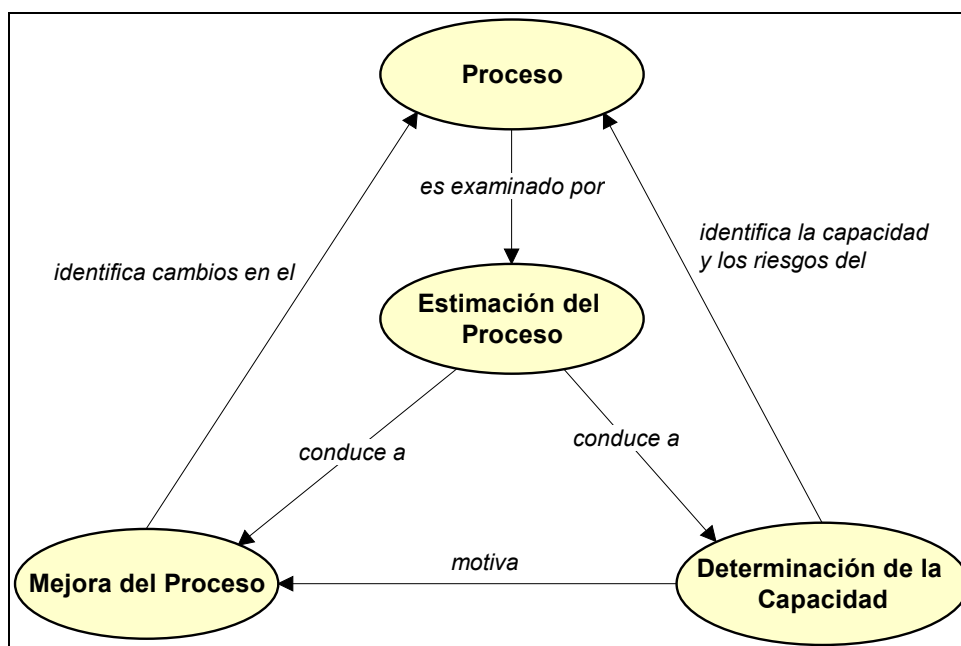


Figura 3-19. Contexto de la estimación de procesos software.

De los dos contextos citados para la estimación de procesos, en este trabajo nos centramos únicamente en el de mejora, que es el que tiene relación más directa con el modelado de procesos software. En este contexto, la estimación de procesos provee un medio de caracterizar las prácticas actuales de una unidad organizacional en términos de capacidad de los procesos seleccionados. El análisis de los resultados de la estimación en relación con las necesidades de negocio de la organización ayuda a identificar fortalezas, debilidades y riesgos inherentes a los procesos. Esto, a su vez, posibilita determinar si los procesos son eficientes logrando sus objetivos, e identificar las causas más significativas de la falta de calidad o los excesos en tiempo y/o costes. Por último, conocidas dichas causas, se pueden determinar los

cambios necesarios en el proceso propiamente dicho (y en consecuencia, en su modelo correspondiente) para mejorarlo.

El modelo de referencia de procesos propuesto en ISO 15504-2 (ISO/IEC, 1998c) tiene una arquitectura con dos dimensiones:

- La dimensión del proceso, caracterizada por descripciones del propósito de los procesos, las cuales, en esencia, son objetivos medibles de los procesos. Por tanto, define los procesos que pueden ser estimados.
- La dimensión de la capacidad de proceso, caracterizada por una serie de atributos del proceso, aplicables a cualquier proceso, que representan las características medibles necesarias para gestionar un proceso y mejorar la capacidad de realizarlo. Es decir, define la escala para medir las capacidades de los procesos.

Cada proceso en este modelo de referencia es definido por una descripción de propósito. Estas descripciones incluyen los objetivos funcionales del proceso cuando es instanciado en un entorno particular. También incluyen material adicional identificando los resultados que se obtienen al aplicar con éxito cada proceso. El modelo de referencia está basado en la norma ISO 12207 ya presentada, a la cual completa y mejora en algunos aspectos. De esta forma, en la dimensión del proceso se considera que los procesos del ciclo de vida del software se dividen en los mismos tres grupos que en ISO 12207 y, además, se establecen cinco categorías en las que se subdividen:

1. Los **procesos primarios** están divididos en dos categorías:
 - Cliente-Suministrador (CUS), con los procesos que afectan directamente al cliente y permiten la operación y uso adecuados del producto y/o servicio.
 - Ingeniería (ENG), que consiste en los procesos que directamente especifican, implementan o mantienen el producto software, su relación con el sistema y su documentación.
2. Los **procesos de soporte** forman una única categoría de igual nombre:
 - Soporte (SUP), formada por los procesos que pueden ser empleados por cualquiera de los otros procesos (incluidos otros procesos de soporte) en diferentes momentos del ciclo de vida del software.
3. Los **procesos organizacionales** se dividen en dos categorías:
 - Gestión (MAN), que engloba a los procesos que contienen prácticas de carácter general que pueden usarse por cualquiera que gestiona cualquier tipo de proyecto o proceso dentro del ciclo de vida de un producto software.
 - Organización (ORG), formada por los procesos que establecen los objetivos de negocio de la organización y desarrollan los activos (de proceso, de producto o recursos) que ayudan a la organización a alcanzar dichos objetivos de negocio cuando son utilizados en los proyectos que lleva a cabo.

El modelo de referencia no define cómo, o en qué orden, deben conseguirse los objetivos de un proceso. Estos objetivos se alcanzan a través de diversas actividades detalladas, tareas y prácticas que deben ser llevadas a cabo para producir los productos de trabajo. La realización de estas actividades, tareas, y prácticas, junto con las características de los productos

de trabajo obtenidos, demostrarán cuando está siendo alcanzado el propósito de un proceso específico.

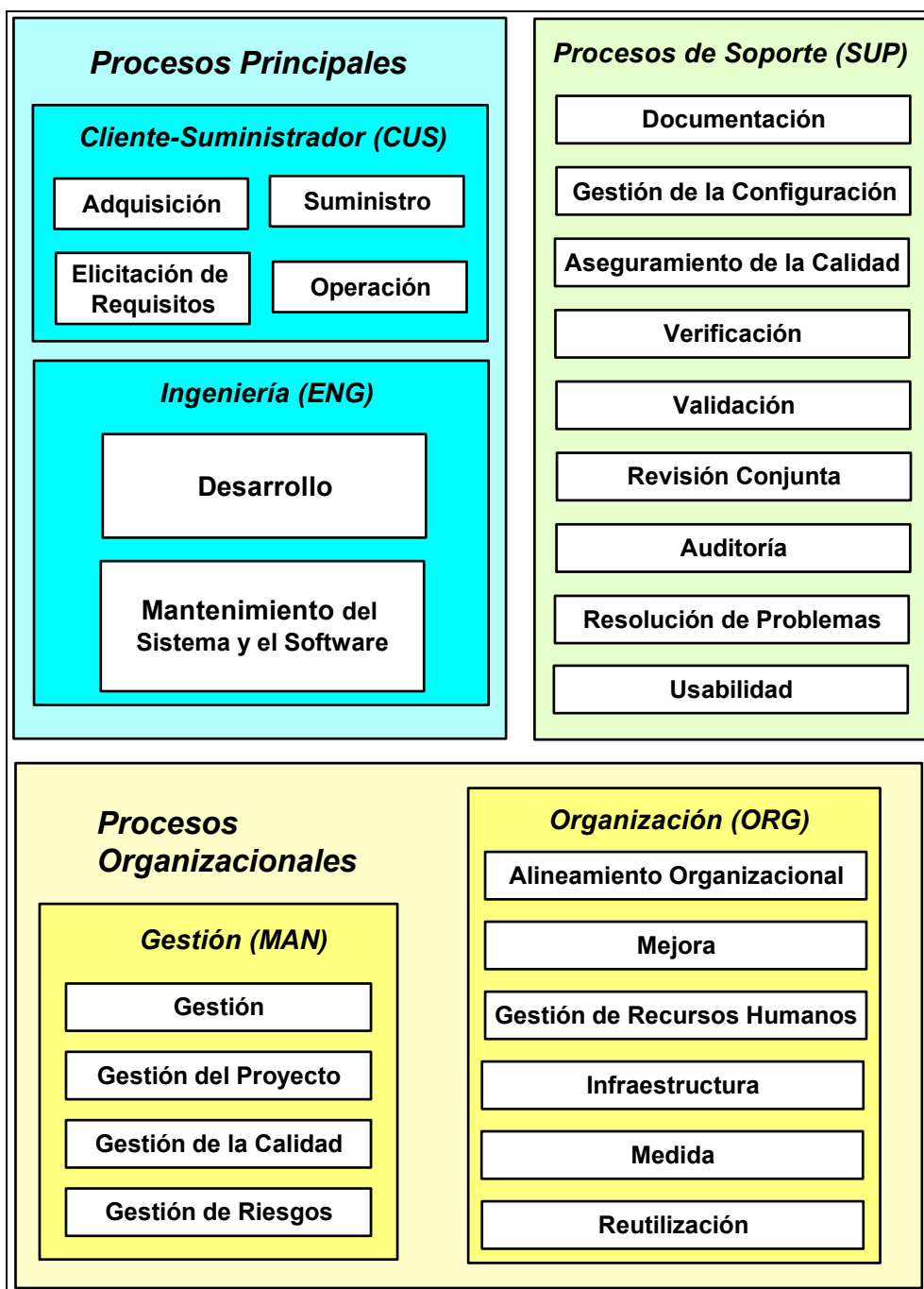


Figura 3-20. Categorías y procesos en el modelo de referencia ISO 15504.

Para detectar fácilmente las similitudes y diferencias con ISO 12207, en la Figura 3-20 se muestran estos procesos, categorías y grupos utilizando un esquema similar al de la Figura 3-17. La principal diferencia, ya comentada, consiste en la existencia de dos niveles de procesos (grupos y categorías) frente a únicamente grupos en ISO 12207. La segunda diferencia

importante es la existencia de los siguientes nuevos procesos o cambios en procesos de ISO 12207:

- Elicitación de Requisitos (nuevo, principal/cliente-suministrador): El propósito de este proceso es recopilar, procesar y hacer un seguimiento de la evolución de las necesidades y requisitos del cliente a lo largo de la vida del producto o servicio software, así como establecer una línea base (*baseline*) de requisitos que sirva como fundamento para definir los productos de trabajo necesarios.
- Gestión del Proyecto (nuevo, organizacional/gestión): Su objetivo es identificar, establecer, coordinar y supervisar las actividades, tareas y recursos necesarios en un proyecto para producir un producto o servicio que cumpla los requisitos.
- Gestión de la Calidad (nuevo, organizacional/gestión): Este proceso busca supervisar la calidad de los productos y/o servicios de un proyecto y asegurar que satisfacen al cliente. El interés se debe centrar en la calidad de los productos y procesos, tanto a nivel de proyecto como de la organización en su conjunto.
- Gestión de Riesgos (nuevo, organizacional/gestión): El propósito es identificar y mitigar los riesgos de un proyecto de forma continua a lo largo del ciclo de vida de un proyecto. Para ello se debe realizar una supervisión de los riesgos a nivel de proyecto y a nivel de la organización en general.
- Alineamiento Organizacional (nuevo, organizacional/organización): Su objetivo es asegurar que los individuos de la organización comparten una visión, cultura y conocimiento de los objetivos del negocio comunes, que les faculte para funcionar de forma eficiente.
- Medida (nuevo, organizacional/organización): El propósito de este proceso es recoger y analizar datos relativos a los productos desarrollados y a los procesos implementados por una unidad organizacional, para ayudar a la gestión eficaz de los procesos y a la demostración objetiva de la calidad de los productos.
- Mantenimiento del Sistema y del Software (cambio, principal/ingeniería): Mientras que en ISO 12207 este proceso se refiere únicamente al mantenimiento del software, ahora incluye también el hardware, redes, etc.
- Reutilización (cambio, organizacional/organización): Equivale al proceso de Gestión de la Reutilización en ISO 12207 cambiando su objetivo. Ahora el propósito es promover y facilitar la reutilización de productos de trabajo software desde una perspectiva organizacional y de producto/proyecto.
- Gestión (cambio, organizacional/gestión): El proceso de Gestión incluido en ISO 12207 ha sido dividido en varios: Gestión (en general), Gestión del Proyecto, Gestión de la Calidad, Gestión de Riesgos, Alineamiento Organizacional y Medida.
- Los procesos de Usabilidad, Gestión de Activos e Ingeniería de Dominios no están incluidos en la versión actual de ISO 15504 porque se incluyeron en la mejora de ISO 12207 elaborada con posterioridad. Cabe suponer que en la reforma de ISO 15504, en realización en la actualidad, se producirá una convergencia del modelo de procesos entre ambas normas corrigiendo estas divergencias.

Otra diferencia menor entre ambas normas tiene que ver con la nomenclatura utilizada. ISO 15504 establece dos niveles jerárquicos de procesos, de forma que algunos procesos (llamados básicos) están formados por otros procesos más simples (llamados componentes). En cambio, en ISO 12207 no existen procesos componentes sino actividades.

El documento de mejora de ISO 15504-2 (ISO/IEC, 2001a) se dedica únicamente a conceder más importancia al rol de adquisidor de software. Para ello propone que el proceso de Adquisición pase a ser una categoría dentro de los procesos principales formada por 17 procesos diferentes, que se corresponden con las actividades y/o tareas del proceso original.

3.3.3. El Proceso de Mantenimiento: ISO 12207 y 14764.

Como hemos visto anteriormente, la norma ISO 12207 considera el mantenimiento como uno de los procesos principales del ciclo de vida del software, que define las actividades y tareas del mantenedor (la organización que proporciona el servicio de mantener el producto software). A continuación se presenta el modelo de PMS establecido a grandes rasgos en ISO 12207 y detallado en el documento borrador del nuevo estándar ISO 14764 (ISO/IEC, 1998e). Al igual que pasa con los demás estándares, en estos documentos únicamente se especifica cuáles deben ser las actividades y tareas del proceso de mantenimiento pero no se indica cómo realizarlas.

En ISO 14764 se considera que el PMS se activa cuando se modifica un producto software después de su entrega. Se distinguen cuatro **tipos de mantenimiento** diferentes según los objetivos de dicha modificación sean:

- Corregir problemas detectados (Correctivo).
- Mejorar la funcionalidad, o el rendimiento, mantenibilidad u otros atributos del software (Perfectivo).
- Hacer que siga siendo utilizable en un entorno nuevo o cambiado (Adaptativo).
- Detectar y corregir errores latentes en el software antes de que se conviertan en fallos reales (Preventivo).

Cuando el mantenedor recibe una **petición de modificación** (propuesta de cambio de un producto software que está siendo mantenido), según su naturaleza, implicará una corrección o una mejora en el software. Una corrección supone un mantenimiento de tipo correctivo, mientras que una mejora puede suponer un mantenimiento preventivo, adaptativo o perfectivo (Figura 3-21).

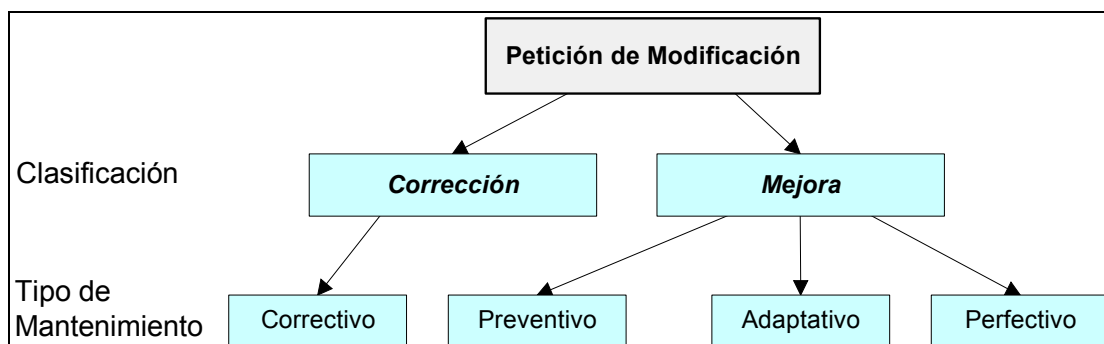


Figura 3-21. Peticiones de modificación y tipos de mantenimiento.

3.3.3.1. Actividades y Tareas.

En la definición de ISO (ISO/IEC, 1995 y 1998e), el PMS consta de las seis actividades siguientes (Figura 3-22):

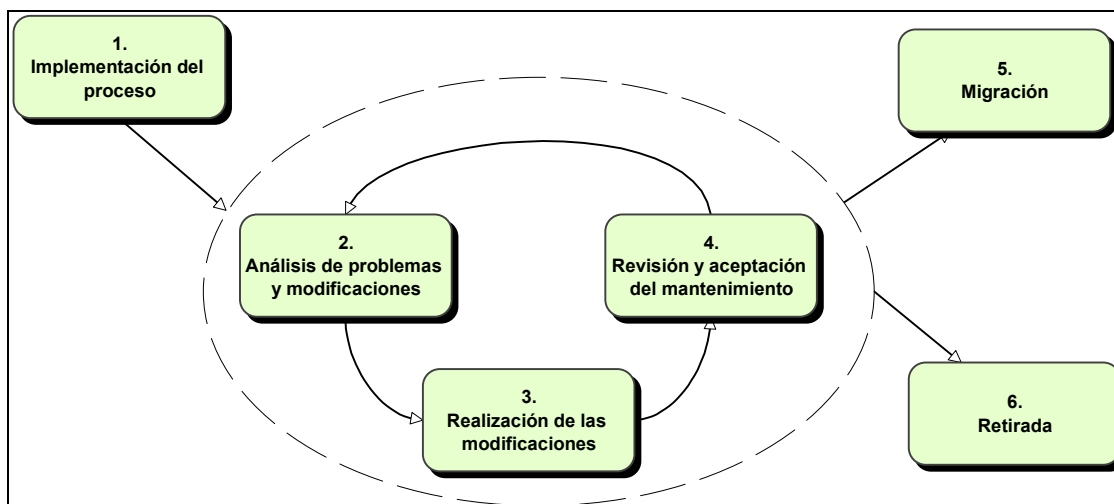


Figura 3-22. El proceso de mantenimiento según ISO.

A1: Implementación del proceso.

En esta actividad se desarrollan los planes (por ejemplo, el plan de mantenimiento) y procedimientos correspondientes para llevar a cabo las actividades y tareas del PMS. También se deben definir los procedimientos necesarios para la gestión de los problemas (empleando el proceso de resolución de problemas) y de las peticiones de modificación, e implementar el proceso de gestión de la configuración para gestionar los cambios en los elementos software existentes. Consta de tres tareas:

1. Elaborar, documentar y ejecutar planes y procedimientos para conducir las actividades y tareas del proceso.
2. Establecer procedimientos para recibir, almacenar y controlar los informes de problemas y solicitudes de modificación de los usuarios, y proporcionar a éstos realimentación.
3. Implementar (o establecer una interfaz organizacional con) el proceso de Gestión de la Configuración, para gestionar las modificaciones del sistema existente.

A2: Análisis de problemas y modificaciones.

Antes de modificar el software mantenido, el mantenedor deberá analizar los problemas o peticiones de modificación, elaborar y documentar distintas soluciones potenciales, y obtener la aprobación para implementar la opción elegida. En consecuencia, durante esta actividad, el mantenedor realiza las siguientes tareas:

4. Analizar el impacto de la petición o informe en la organización, el sistema existente y los interfaces con otros sistemas, determinando el tipo de mantenimiento (uno de los cuatro anteriores), el alcance (tamaño, coste y tiempo

para hacer la modificación) y su criticidad (impacto en el rendimiento, la seguridad, etc.).

5. Replicar o verificar el problema.
6. Elaborar opciones para implementar la modificación.
7. Documentar el informe o petición, los resultados del análisis, y las opciones de implementación.
8. Obtener aprobación para la opción de modificación elegida.

A3: Realización de las modificaciones.

En esta actividad se incluyen todas las tareas relativas a determinar qué elementos del sistema software (documentación, unidades o módulos y versiones) deben modificarse, y se utiliza el proceso de desarrollo para implementar las modificaciones. Las tareas que lleva a cabo el mantenedor durante esta actividad son las siguientes:

9. Realizar un análisis para determinar qué documentación, unidades de software y versiones necesitan ser modificadas. Documentar esta información.
10. Entrar en el proceso de Desarrollo para implementar las modificaciones. Este proceso, según es definido en ISO 12207 (ISO/IEC, 1995), debe complementarse con lo siguiente:
 - a) definir y documentar criterios para probar y evaluar las partes del sistema modificadas y no modificadas;
 - b) comprobar la implementación completa y correcta de los requisitos nuevos o modificados;
 - c) asegurarse de que los requisitos originales no modificados no han sido afectados; y
 - d) documentar los resultados de estas pruebas.

A4: Revisión y aceptación del mantenimiento.

En esta actividad se asegura que las modificaciones al sistema son correctas y que cumplen los estándares aprobados utilizando una metodología correcta. Las dos tareas que lleva a cabo el mantenedor durante esta actividad son:

11. Realizar revisiones conjuntas entre el mantenedor y la organización que debe aprobar la modificación para determinar la integridad del sistema.
12. Obtener la aprobación, según los términos indicados en el contrato o acuerdo de mantenimiento, de que la modificación se ha completado satisfactoriamente.

A5: Migración.

Esta actividad se realiza cuando el sistema tiene que ser modificado para ser ejecutado en un entorno diferente. En ese caso, el mantenedor necesita determinar las acciones necesarias para lograr la migración, llevarlas a cabo, y documentar los efectos. Las tareas incluidas en esta actividad son las siguientes:

13. Asegurar que cualquier producto software o dato producido o modificado durante la migración lo ha sido de conformidad a lo establecido en la norma ISO 12207.

14. Elaborar y documentar un “plan de migración” en el que se incluirán, al menos, las siguientes cuestiones:
 - Análisis y definición de los requisitos de la migración.
 - Herramientas para la migración.
 - Conversiones del software y de los datos.
 - Ejecución de la migración.
 - Verificación de la migración.
 - Soporte futuro del entorno antiguo.
15. Notificar a los usuarios las intenciones de migración previstas, indicando lo siguiente:
 - a) explicación de porqué no se puede seguir soportando el entorno antiguo;
 - b) descripción del nuevo entorno, indicando la fecha de disponibilidad prevista; y
 - c) descripción de otras opciones de soporte cuando el entorno antiguo haya sido eliminado.
16. Realizar operaciones en paralelo en ambos entornos, el viejo y el nuevo, para que la transición sea suave. A la vez, proveer a los usuarios de la formación necesaria.
17. Notificar a todos los afectados de que se ha completado la migración. Toda la documentación, elementos del sistema y datos del entorno antiguo deberá ser recopilada y archivada.
18. Realizar una revisión post-operación para evaluar el impacto del cambio al entorno nuevo. Los resultados serán enviados a las autoridades adecuadas.
19. Establecer la accesibilidad a los datos utilizados o asociados con el entorno antiguo de acuerdo con los requisitos del contrato en cuanto a protección de datos y auditoría.

A6: Retirada.

Esta actividad se realiza cuando un producto software ha alcanzado el final de su vida útil. Para ello, se deben llevar a cabo las siguientes tareas:

20. Elaborar y documentar un “plan de retirada”, incluyendo los siguientes apartados:
 - Cese total o parcial del soporte después de un cierto tiempo.
 - Archivo del producto software y su documentación asociada.
 - Responsabilidad sobre cuestiones de soporte residual futuro.
 - Transición al nuevo producto, si lo hubiera.
 - Verificación de la migración.
 - Accesibilidad a las copias de los datos archivados.
21. Notificar a los usuarios las intenciones de retirada previstas, indicando lo siguiente:
 - a) descripción de la sustitución o actualización previstas, indicando la fecha de disponibilidad;

- b) explicación de porqué no se puede seguir soportando el producto retirado; y
 - c) descripción de otras opciones de soporte disponibles una vez el producto se haya retirado.
22. Realizar operaciones para una transición suave a la nueva situación, incluyendo la formación necesaria a los usuarios.
 23. Notificar a todos los afectados de que se ha completado la retirada. Toda la documentación, elementos del sistema y datos deberá ser recopilada y archivada.
 24. Establecer la accesibilidad a los datos utilizados o asociados con el producto retirado de acuerdo con los requisitos del contrato en cuanto a protección de datos y auditoría.

Las actividades de implementación del proceso y de retirada se deben realizar obligatoriamente una sola vez por parte del mantenedor al inicio y final del servicio o proceso de mantenimiento. La actividad de migración podrá ocurrir varias veces o ninguna en momentos aleatorios, según se produzcan cambios en el entorno o no (sistema operativo, SGBD, red, hardware, etc.). Las tres actividades intermedias ocurren una vez por cada petición de modificación o informe de problemas recibido por el mantenedor. Estas tres actividades forman un ciclo en el sentido de que el resultado de una revisión y aceptación del mantenimiento puede conducir a nuevas peticiones de modificación o informes de problemas.

Proceso de Mantenimiento del Software		
Actividades	Tareas	
Implementación del proceso	1	Desarrollar planes y procedimientos de mantenimiento
	2	Establecer procedimientos para peticiones de modificación
	3	Implementar la gestión de la configuración
Análisis de problemas y modificaciones	4	Analizar el informe o petición
	5	Replicar o verificar el problema
	6	Elaborar opciones para implementar la modificación
	7	Documentar
	8	Obtener aprobación para la opción elegida
Realización de las modificaciones	9	Determinar los elementos a modificar
	10	Invocar al proceso de desarrollo
Revisión y aceptación del mantenimiento	11	Revisar la integridad del sistema modificado
	12	Obtener aprobación
Migración	13	Asegurar adaptación a las normas
	14	Elaborar plan de migración
	15	Notificar intenciones a los usuarios
	16	Ejecutar operaciones en paralelo y formar a los usuarios
	17	Notificar la compleción de la migración
	18	Realizar revisión post-operación
	19	Archivar datos del entorno antiguo
Retirada	20	Desarrollar el plan de retirada
	21	Notificar futura retirada
	22	Ejecutar en paralelo
	23	Notificar retirada
	24	Archivar datos del entorno antiguo

Tabla 3-5. Actividades y tareas del proceso de mantenimiento según ISO.

En la Tabla 3-5 se muestra un resumen de todas las actividades y tareas relatadas anteriormente.

Aquel que ignora la historia está condenado a repetirla.

Jorge Santayana en "The Life of Reason" (1905-6).

4. Gestión del Mantenimiento

En este capítulo se aborda el problema de la gestión del PMS. Para ello, primero se presenta un estudio bibliográfico que ha permitido sacar interesantes conclusiones sobre la situación de la investigación en este tema en los últimos años. A continuación se incluyen diversos apartados que abarcan las distintas dimensiones implicadas en la gestión del PMS: modelos y métodos para la gestión global del proceso, gestión del proyecto, gestión de la calidad, gestión de riesgos, mejora del proceso, gestión de recursos humanos, infraestructura (herramientas), medición del proceso, reutilización, y alineamiento con los objetivos de la organización.

4.1. Introducción.

En general, el concepto de Gestión es muy amplio. Gestión “son todas las actividades y tareas realizadas por una o mas personas con el propósito de planificar y controlar las actividades de otras personas, en orden a lograr un objetivo o completar una actividad que no podría conseguirse por esas otras personas actuando solas” (Thayer, 1997). De esta definición se concluye que la coordinación es la función esencial de la gestión, entendiendo que “coordinación es la gestión de las dependencias entre actividades” (Malone y Crowston, 1994).

En conclusión, en el ámbito de esta tesis podemos resumir que la *Gestión del Proceso de Mantenimiento del Software es la coordinación del trabajo de los ingenieros que cooperan para mantener un producto software*.

Por otro lado, la gestión es una actividad compleja y multidimensional. En este sentido, Westfechtel (1999b) ha propuesto un marco conceptual de la gestión de procesos de desarrollo, que establece 4 dimensiones de gestión diferentes:

- *Objetos*: los objetos a gestionar se clasifican en productos, actividades y recursos. Estos objetos se combinan en configuraciones técnicas o de gestión.
- *Granularidad*: Una configuración técnica representa los objetos (productos, actividades y recursos) en detalle (granularidad fina). En cambio, una configuración de gestión es una abstracción de la anterior y representa a los objetos con granularidad media o gorda.
- *Control*: Se distingue entre actividades técnicas y de gestión, de forma que las primeras son controladas por las segundas.
- *Modelado*: Se distinguen distintos niveles de modelado, que básicamente coinciden con los niveles de modelado comentados en el apartado 3.1.4.3. Una “instancia de proceso” representa un proceso concreto. Una “definición de proceso” representa una clase de procesos, que se concreta en instancias de proceso. Un “metamodelo de proceso” es una colección de facilidades para crear definiciones de proceso.

Hace ya bastantes años que existe bibliografía dedicada a la gestión de los procesos de ingeniería del software en general (Humphrey, 1990), pero la atención a la gestión de los PS desde un punto de vista más formal y automatizado es más reciente (Derniame et al, 1999a), y además, ha estado centrada casi de forma total en los procesos de desarrollo (Westfechtel, 1999a), dejando en un segundo plano al proceso de mantenimiento.

Para tener una visión más afinada de la situación de la investigación en gestión del mantenimiento de software, se ha procedido a realizar un estudio bibliográfico que se presenta en el próximo apartado. De forma complementaria, en los restantes apartados, se resumen algunas de las principales aportaciones en esta área. También se puede consultar el libro “*Advances in Software Maintenance Management: Technologies and Solutions*” (Polo et al, 2003), cuyos editores han sido varios profesores del grupo Alarcos (incluido el autor de esta tesis) y que incluye capítulos de algunos de los más conocidos expertos internacionales en mantenimiento de software.

4.2. Estudio Bibliográfico.

Tradicionalmente, el PMS ha estado en un segundo plano en las actividades de I+D en ingeniería del software. Además, los aspectos de gestión dentro del mantenimiento han recibido una atención bastante minoritaria. Confirmar estadísticamente estas dos afirmaciones es el origen del estudio de fuentes bibliográficas que se presenta a continuación.

El objetivo concreto del estudio ha sido encontrar todas⁹ las referencias científicas (artículos y ponencias) relacionadas con la gestión del mantenimiento de software y realizar un pequeño análisis estadístico que permita sacar conclusiones sobre la situación de la investigación en esta materia y la atención que le ha prestado la comunidad científica internacional.

4.2.1. Método Empleado.

El trabajo de búsqueda, consulta y clasificación de las referencias se ha realizado durante los meses de mayo-julio de 2002; y se han establecido las siguientes consideraciones:

- 1) Sólo se han consultado publicaciones de los años 1998, 1999, 2000 y 2001. No se han tenido en cuenta años anteriores para garantizar la actualidad de los resultados. Se ha considerado que 4 años es un tiempo adecuado para evitar posibles “factores temporales” que desvirtúen los resultados. No se han considerado los primeros meses del año 2002 porque entonces no se cumpliría la homogeneidad temporal, en el sentido de que algunos números de revistas o libros de actas ya habrían sido publicados y otros todavía no.
- 2) Se han utilizado 5 tipos de medios diferentes en las búsquedas:
 - *Publicaciones especializadas*. Se han consultado en soporte papel todos los artículos y ponencias aparecidos en las tres principales publicaciones mundiales dedicadas al mantenimiento de software: “*Journal of Software Maintenance and Evolution*” (JSME), que hasta el año 2000 se llamó “*Journal of Software Maintenance - Research and Practice*”; “*International Conference on Software Maintenance*” (ICSM); y “*European Conference on Software Maintenance and Reengineering*” (CSMR).
 - *Revistas y conferencias publicadas o patrocinadas por ACM e IEEE-CS*. Se han consultado los documentos electrónicos de todos los artículos y ponencias relacionados con la gestión del PMS. Para ello se han utilizado las suscripciones disponibles a sendas bibliotecas digitales (ACM, 2002b), e (IEEE-CS, 2002).
 - *Conferencias cuyos libros de actas pertenecen a la colección “Lecture Notes in Computer Science”*. Se han podido obtener en soporte papel o digital alrededor del 80% de los artículos y ponencias relacionados con el PMS. Del restante 20% se ha consultado su resumen (*abstract*). Las referencias de interés fueron localizadas utilizando las facilidades de búsqueda del servidor web de la editorial, Springer-Verlag (2002).

⁹ Se refiere a “todas” las incluidas en las publicaciones más relevantes sobre el tema.

- *Revistas publicadas por la editorial Kluwer.* Se utilizó el servicio de búsqueda en artículos provisto por la propia editorial (Kluwer, 2002). Las referencias de interés fueron clasificadas analizando sus resúmenes.
 - *Otras revistas y conferencias.* La localización de las referencias de interés se ha llevado a cabo utilizando varios servicios web de bibliografía científica: “*Citeseer Scientific Literature Digital Library*” (NEC, 2002); “*DBLP Computer Science Bibliography*” (DBLP, 2002); y “*The Collection of Computer Science Bibliographies*” (CSB, 2002). Una vez localizada una referencia, se ha recuperado directamente el documento electrónico si estaba disponible (este servicio sólo es ofrecido por CiteSeer y CSB pero no para todas sus referencias) o se ha procedido a su búsqueda en la web utilizando un buscador general (*Google*). De esta forma se han conseguido recuperar alrededor del 60% de los artículos y ponencias. Otro 30% ha tenido que ser analizado únicamente a través del resumen mostrado por los servicios bibliográficos citados. Sólo en un 5% de las referencias se ha tenido que recurrir exclusivamente al título.
- 3) Las referencias encontradas han sido clasificadas en base a la dimensión de la gestión del PMS a la que se refieren. Cuando una referencia abordaba más de una dimensión se ha clasificado en la que se ha considerado que hace una aportación más significativa. Estas dimensiones han sido establecidas a partir del modelo de referencia ISO 15504 (ISO/IEC, 1998c) para procesos del ciclo de vida del software (ver apartado 3.3.2), incluyendo los 10 procesos organizacionales que indica dicha norma. También se ha incluido otra dimensión, de carácter general, referida a los modelos y métodos para gestionar el PMS. De esta forma, las dimensiones de gestión definidas han sido las siguientes (entre paréntesis se indica la abreviatura utilizada después):
- Gestión global del PMS (GES).
 - Gestión del proyecto - planificación, seguimiento, control (PRO).
 - Gestión de la calidad (CAL).
 - Gestión de riesgos (RIE).
 - Alineamiento organizacional (ALI).
 - Mejora y evaluación del proceso (MEJ).
 - Gestión de recursos humanos (HUM).
 - Infraestructura - herramientas y entornos (INF).
 - Medida - métricas (MED).
 - Reutilización (REU).
 - Modelos y métodos para la gestión del PMS (MOD).

4.2.1.1. Acotamiento de las Búsquedas.

La manera de acotar las búsquedas ha sido diferente en función del tipo de medio. Los principales datos al respecto son los siguientes:

- a) Las tres publicaciones especializadas en mantenimiento del software consultadas (CSMR, ICSM y JSME) incluyen un total de 374 artículos y ponencias durante los años contemplados en el estudio. De dichas referencias, únicamente 32 abordan aspectos de gestión (ver Tabla 4-1).

Referencias en publicaciones especializadas por fuente y año						
Fuente	1998	1999	2000	2001	Total	NRG¹⁰
CSMR	36	24	22	19	101	8
ICSM	38	52	30	79	199	18
JSME	18	19	17	20	74	6
TOTAL	92	95	69	118	374	32

Tabla 4-1. Artículos y ponencias en publicaciones especializadas en mantenimiento.

- b) En cuanto a las revistas y actas publicadas por ACM, se ha realizado una búsqueda en el portal digital (ACM, 2002b) con la palabra clave “maintenance” en el título o en el resumen. El resultado han sido 137 referencias entre los años 1998 y 2001. De ellas, tan sólo 3 han abordado asuntos de interés para la gestión del PMS: 2 en la conferencia ICSE y 1 en la revista SEN (consultar el anexo A para saber el significado de las siglas de las fuentes).
- c) En el caso de publicaciones patrocinadas por IEEE-CS, la búsqueda se realizó en la biblioteca digital (IEEE-CS, 2002) con la palabra clave “maintenance” en el título (no existe la posibilidad de restringir por el resumen). El resultado fueron 100 referencias, de las cuales 31 eran de la conferencia ICSM, 18 de la conferencia CSMR y 2 de ICSE, que ya están consideradas anteriormente; quedando, por tanto, 49 referencias a analizar. De ellas, las relacionadas con aspectos de gestión del mantenimiento fueron 6: 2 en la revista IEEE-Software, 1 en la revista TSE y 3, una por evento, en las conferencias APSEC, ISESS e IWPC.
- d) Las actas de conferencias publicadas en la colección LNCS se acotaron mediante una búsqueda en el servidor de la editorial que las publica (Springer-Verlag, 2002), obteniéndose un total de 40 referencias que incluían las palabras “maintenance” y “software” en el resumen. En este caso se optó por añadir la palabra software porque el servicio de búsqueda de la editorial es general para todos los campos científicos y no está restringido sólo a la colección LNCS. Al comprobar que algunas de las conferencias no incluían algunas de sus ediciones (de los años 1998-2001) en dicha colección, se completó la búsqueda consultando directamente los programas finales en la web. En total, una vez excluidas las referencias ya obtenidas en los pasos anteriores, sólo quedaron 2 ponencias sobre la gestión del PMS, ambas pertenecientes a la conferencia PROFES. Entre las varias docenas de conferencias publicadas en la colección LNCS, en la Tabla 4-2 se muestra la lista de las que, por su temática, fueron revisadas con más cuidado, ponencia a ponencia; indicando, en su caso, la localización en la web o el número de la colección LNCS. Es de destacar que en la conferencia EWSPT, dedicada a la tecnología de proceso software (incluidos los EIS), no se encontró ninguna referencia orientada al mantenimiento.

¹⁰ Número de referencias que abordan aspectos de gestión del PMS.

Conferencias de la colección LNCS consultadas			
Fuente	Año ¹¹	NRG	Localización
Int. Conf. on Product Focused Software Process Improvement (PROFES)	2001	1	LNCS 2188
	2000		LNCS 1840
	1999	1	http://www.vtt.fi/ele/profes99/
European Workshop on Software Process Technology (EWSPT)	2001		LNCS 2077
	2000		LNCS 1780
	1998		LNCS 1487
Conference on Advanced Information Systems Engineering (CAiSE)	2001		LNCS 2068
	2000		LNCS 1789
	1999		LNCS 1626
	1998		LNCS 1413
Int. Conf. on Fundamental Approaches to Software Engineering (FASE)	2001		LNCS 2029
	2000		LNCS 1783
	1999		LNCS 1577
	1998		LNCS 1382
Int. Workshop on Software Measurement (IWSM)	2001		http://www.lrgl.uqam.ca/iwsm2001/
	2000		LNCS 2006
	1999		http://www.lrgl.uqam.ca/iwsm99/index2.html
	1998		http://irb.cs.uni-magdeburg.de/sw-eng/us/workshop/
Int. Conf. on Software Reuse (ICSR)	2000		LNCS 1844
European Software Engineering Conf. (ESEC)	2001		http://esec.ocg.at/
	1999		LNCS 1687
TOTAL		2	

Tabla 4-2. Conferencias publicadas en la colección LNCS que han sido consultadas en detalle.

- e) En las revistas de la editorial Kluwer se buscaron referencias que tuvieran las palabras “maintenance” y “software” en el resumen. De esta manera se obtuvieron un total de 69, de las cuales 35 pertenecían a los años 1998-2001. Del análisis de dichas referencias se obtuvo una referencia dedicada a la gestión del PMS, que fue publicada en la revista “*Annals of Software Engineering*”. Es de destacar que se obtuvieron 5 referencias de interés adicionales, pero publicadas en el año 2002, 4 en la revista “*Information Technology and Management*” (ITM), y 1 en “*Empirical Software Engineering*” (ESE).
- f) Las búsquedas en CiteSeer (NEC, 2002) no permiten restringir por años ni por campos (hace una indexación completa de los documentos en formato Postscript). Por esta razón, primero se obtuvieron 2683 referencias que incluían las palabras “maintenance” y “software”. Mediante una restricción para que dichas palabras estuvieran en el “header” (del documento Postscript) ese número tan alto se redujo a 178. La revisión a mano de dichas referencias redujo el número a 40 entre los años 1998 y 2001. Por último, del análisis de estas 40 referencias no se obtuvo ninguna, relacionada con la gestión del PMS, que no estuviera repetida.
- g) El servicio de búsquedas DBLP (2002) sólo permite buscar por autor(es) o título. En consecuencia, se obtuvieron 201 referencias que incluían la palabra “maintenance” en el título. No se añadió la palabra “software” porque este servicio, al contrario que CiteSeer, está especializado en Informática. La revisión manual redujo las referencias a 73 en los 4 años incluidos en el estudio, que se quedaron en 49 al eliminar las ya

¹¹ Sólo se han incluido los años en que hubo evento.

obtenidas en las búsquedas previas. De estas últimas, sólo 1 ponencia, perteneciente a la conferencia ICEIS, trataba aspectos de gestión del mantenimiento.

- h) El servicio CSB permite añadir restricciones por bastantes parámetros. Por ello, se realizó una búsqueda que incluyera las palabras “maintenance” y “software” en cualquier campo, con la restricción de años ya conocida, y se obtuvieron 109 referencias para documentos de tipo “Journal Article” y 258 para los de tipo “Conference Paper”. En total se recuperaron 367 referencias. Mediante una costosa revisión manual de los títulos y resúmenes (obtenidos con la opción Presentation=“Full References”), se comprobó que tan sólo 26 referencias hablaban de gestión del PMS, pero únicamente 3 de ellas no se habían obtenido antes en otras búsquedas. Estas 3 referencias nuevas eran todas de la revista JSS.
- i) Adicionalmente, se procedió a buscar en las ponencias de 2 conferencias que se consideró que podían aportar alguna referencia adicional: ICSE e IWPSE. Se consultaron los programas finales (obtenidos de las direcciones web indicadas en la Tabla 4-3), pero las 2 ponencias (de ICSE), relacionadas con la gestión del mantenimiento, que se encontraron estaban repetidas.

Otras conferencias consultadas		
Fuente	Año	Localización
Int. Conf. on Software Engineering (ICSE)	2001	http://www.csr.uvic.ca/icse2001/
	2000	http://www.ul.ie/~icse2000/
	1999	http://sunset.usc.edu/icse99/
	1998	http://holstein.aist-nara.ac.jp/icse98/
Int. Workshop on Principles of Software Evolution (IWPSE)	2001	http://www.jaist.ac.jp/IWPSE2001/
	1999	http://dontaku.csce.kyushu-u.ac.jp/IWPSE99/
TOTAL		

Tabla 4-3. Otras conferencias cuyos programas finales fueron consultados.

- j) Por último, puesto que en el servicio de búsqueda CSB se encontraron 3 artículos de la revista JSS, se pensó que esta revista podría incluir alguna referencia más de interés para el estudio. Por ello, se utilizó el servicio especial de búsqueda web facilitado por la editorial (Elsevier, 2002). De esta manera se obtuvieron 29 referencias que incluían las palabras “maintenance” y “software” en su resumen. Analizadas las 29, únicamente las 3 ya conocidas tenían relación con la gestión del mantenimiento.

4.2.2. Resultados y Conclusiones.

En el apartado anterior ya se ha explicado cómo se han ido obteniendo las referencias de interés para el estudio. En total, se han obtenido 48 referencias dedicadas a aspectos de gestión del PMS, cuyos datos detallados (autores, título, fuente, etc.) se muestran en el anexo A. En la Tabla 4-4 se presenta un resumen estadístico de los resultados obtenidos.

Número de referencias por medio, fuente y dimensión de gestión													
Medio	Fuente	GES	PRO	CAL	RIE	ALI	MEJ	HUM	INF	MED	REU	MOD	Total
<i>Pub. especializadas (ESP)</i>	CSMR	1	1				1	1		1		3	8
	ICSM	2	4	1	1		2	1		4		3	18
	JSME			1			1	1	2			1	6
	TOTAL	3	5	2	1		4	3	2	5		7	32
<i>Pub. ACM + IEEE-CS (BD)</i>	APSEC										1		1
	ICSE						1					1	2
	ISESS	1											1
	IWPC								1				1
	SEN							1					1
	SOFT							1		1			2
	TSE									1			1
	TOTAL	1					1	2	1	2	1	1	9
<i>Conf. LNCS (LNCS)</i>	PROFES						1		1				2
	TOTAL						1		1				2
<i>Rev. Kluwer (KLU)</i>	ASE			1									1
	TOTAL			1									1
<i>Otras pub. (OTRAS)</i>	ICEIS											1	1
	JSS			1				1	1				3
	TOTAL			1				1	1			1	4
TOTAL GENERAL		4	5	4	1	0	6	6	5	7	1	9	48

Tabla 4-4. Resumen de resultados del estudio bibliográfico.

Las **principales conclusiones** sacadas de estos resultados son las siguientes:

- En el ámbito del mantenimiento del software, a los aspectos de gestión se les dedica poca atención: tan sólo el 6'2% (48 de 779) de las referencias analizadas (ver Tabla 4-5). En las publicaciones especializadas ha recibido, proporcionalmente, el doble de atención: 8'6% frente a 4'0% al PMS analizadas.
- Se confirma que la principal fuente de consulta en PMS en general, y en los aspectos de gestión de dicho proceso en particular, son las 3 publicaciones especializadas: dos tercios (32 de 48) de las referencias útiles sobre gestión del PMS están publicadas en ellas.
- Las demás publicaciones incluidas en el estudio dedican una atención muy baja al mantenimiento del software. Aunque no se ha realizado el estudio detallado correspondiente, debido a su excesivo coste y dificultad, sirve de prueba observar que en la biblioteca digital de ACM aparecen 17536 referencias sobre "software" durante los años 1998-2001, de las cuales ya hemos visto que sólo 137 (menos del 1%) son sobre mantenimiento. La dedicación a los aspectos de gestión del mantenimiento, tal como se muestra en la Tabla 4-5, es especialmente baja. Otra prueba de esto último es que muchas de las conferencias no han aportado ni una sola referencia de interés para el estudio. Este es el caso de EWSPT, FASE, IWSM, ICSR, ESEC e IWPSE.

Proporción de referencias sobre el PMS que tratan de aspectos de gestión				
Clase de Fuentes	Fuente	Nº Referencias sobre PMS¹²	NRG	%
Especializadas	CSMR	101	8	7.9
	ICSM	199	18	9.0
	JSME	74	6	8.1
	TOTAL	374	32	8.6
No Especializadas	Pub. ACM	137	3	2.2
	Pub. IEEE-CS	49	6	12.2
	Conf. LNCS	40	2	5.0
	Rev. Kluwer	35	1	2.9
	CiteSeer	40	0	0.0
	DBLP	49	1	2.0
	CSB	26	3	11.5
	Elsevier	29	0	0.0
	TOTAL	405	16	4.0
TOTAL GENERAL		779	48	6.2

Tabla 4-5. Importancia asignada a la gestión del mantenimiento.

- Las dimensiones de la gestión más atendidas son MOD (modelos y métodos) y MED (medida) con 9 y 7 referencias respectivamente. Probablemente, las razones son las siguientes: MOD es una dimensión bastante amplia, que realmente engloba aspectos útiles en la gestión pero que son más generales (modelos del proceso, metodologías, tipos de actividades de mantenimiento, etc.); y en la dimensión MED se incluyen las métricas, que han recibido bastante atención, especialmente para estimar los, habitualmente, elevados costes del mantenimiento.
- Las dimensiones de gestión con menos atención han sido ALI (alineamiento organizacional), REU (reutilización) y RIE (gestión de riesgos) con 0, 1 y 1 referencias útiles, respectivamente. Quizás no sean una sorpresa los resultados de las dos primeras porque para el autor del estudio, y parece que para el resto de la comunidad investigadora igual, no queda claro que haya una problemática especial distinta que en otros procesos del ciclo de vida del software, como los de desarrollo. No ocurre lo mismo con la gestión de riesgos, ya que la distinta naturaleza del proceso de mantenimiento hace que sus posibles riesgos sean también diferentes (Pigoski, 1997).

Después de concluido el estudio bibliográfico aquí presentado, el autor tuvo acceso a la base de datos electrónica INSPEC (IEE, 2003). Las búsquedas realizadas con las mismas acotaciones se reseñadas no supusieron ninguna referencia nueva, dedicada a la gestión del mantenimiento, que no hubiera sido ya conocida. Por tanto, esta búsqueda sólo sirvió como confirmación de los resultados anteriores.

Un comentario adicional, ajeno al estudio propiamente dicho, es que 7 de las 48 referencias útiles son del grupo de investigación Alarcos y, por tanto, el autor de esta tesis es coautor.

¹² Excluyendo las repeticiones.

4.3. Propuestas.

En términos financieros, el PMS puede ser visto como un continuo consumidor de recursos, mientras que los beneficios no están claros ni cuantificados (Piattini et al, 2000). Para evitar esta situación se necesita un mayor apoyo por parte de la dirección de las organizaciones para las actividades de mantenimiento, y para ello, es necesario que los gestores veteranos (*seniors*) de las organizaciones sean conscientes de la importancia que las tecnologías de la información tienen para cualquier organización y de que el software es un activo corporativo que puede suponer una ventaja competitiva. Además, los gestores que estén descontentos con la situación y que quieran cambiarla, tendrán que adquirir un compromiso personal y visible con las soluciones de gestión (organizacionales) propuestas.

A continuación se presentan algunas de las propuestas en esta línea. Se han agrupado utilizando la misma clasificación, basada en la norma ISO 15504, que ya hemos presentado en el apartado anterior. Las distintas dimensiones también se presentan en el mismo orden que en el estándar.

4.3.1. Gestión Global del Proceso.

En este apartado incluimos la dimensión del proceso de gestión (general) y también, por su intensa relación con este, las propuestas referidas a modelos y métodos genéricos para gestionar el PMS (que en el estudio bibliográfico hemos incluido en la dimensión MOD).

Es importante hacer una gestión estructurada y organizada del PMS. Este mantenimiento “Estructurado” aparece como resultado de la aplicación de uno o varios métodos de ingeniería del software (Pressman, 1998). Por ejemplo, la existencia de una adecuada “Gestión de la Configuración Software” (documentación e información sobre los requerimientos, especificación, diseño y pruebas) reduce la cantidad de esfuerzo requerido en el mantenimiento y mejora la calidad general de los cambios.

Cuando el mantenimiento no es estructurado, se sufren las consecuencias de la falta de metodología: “dolorosa” evaluación del código (muchas veces poco legible), complicada comprensión del sistema por la pobre documentación interna (desconocimiento de la estructura del programa, las estructuras de datos globales, las interfaces y otros requisitos de diseño y/o rendimiento), dificultad para descubrir las consecuencias de los cambios en el código y, por último, imposibilidad de realizar pruebas de regresión (repetición de pruebas anteriores) al no existir ningún registro de pruebas. Entre las diversas propuestas y métodos para llevar a cabo el mantenimiento de esta forma estructurada, algunas de las más importantes son las siguientes:

4.3.1.1. MÉTRICA 3.

La nueva versión de la metodología MÉTRICA 3 para el ciclo de vida de sistemas de información (CSI, 2000), que contempla el desarrollo de sistemas de información para las distintas tecnologías que actualmente están en uso y los aspectos de gestión que aseguran que un proyecto cumple sus objetivos en términos de calidad, coste y plazos. En esta versión se incluye un capítulo dedicado al Mantenimiento de Sistemas de Información (MSI), al cual se le considera uno de los procesos principales en el ciclo de vida de un sistema de información, junto con los de Planificación y Desarrollo. Aunque desde el enfoque de la norma ISO 15504, el Proceso de Mantenimiento comprende actividades y tareas de modificación o retirada de todos los componentes de un sistema de información (hardware, software, software de base,

operaciones manuales, redes, etc.), este marco de actuación no es el objetivo de MÉTRICA 3, ya que esta metodología está dirigida principalmente al proceso de desarrollo del software. Por ello, MÉTRICA 3 “refleja los aspectos del PMS que tienen relación con el proceso de desarrollo”, es decir, mantenimientos de tipo correctivo o evolutivo (equivalente al preventivo en ISO) fundamentalmente.

En esta metodología, el objetivo del proceso de mantenimiento es la obtención de una nueva versión de un sistema de información desarrollado con MÉTRICA, a partir de las peticiones de mantenimiento que los usuarios realizan con motivo de un problema detectado en el sistema (mantenimiento correctivo) o por la necesidad de una mejora del mismo (preventivo). Ante una petición de cambio de un sistema de información ya en producción, se realiza un registro de las peticiones, se diagnostica el tipo de mantenimiento y se decide si se le da respuesta o no, en función del plan de mantenimiento asociado al sistema afectado por la petición, y se establece con qué prioridad. La definición de la solución al problema o necesidad planteada por el usuario que realiza el responsable de mantenimiento, incluye un estudio del impacto, la valoración del esfuerzo y coste, las actividades y tareas del proceso de desarrollo a realizar y el plan de pruebas de regresión. La lista de actividades y tareas propuestas se muestra en la Tabla 4-6.

En resumen, el PMS definido es bastante parecido al propuesto por ISO (apartado 3.3.3), pero está restringido únicamente a las actividades 2, 3 y 4 del ciclo de mantenimiento de las peticiones de modificación (ver Figura 3-22).

Los productos que se obtienen en este proceso son los siguientes: catálogo de peticiones de cambio; resultado del estudio de la petición; propuesta de solución; análisis de impacto de los cambios; plan de acción para la modificación; plan de pruebas de regresión; evaluación del cambio; y evaluación del resultado de las pruebas de regresión.

El Proceso de Mantenimiento de Sistemas de Información en MÉTRICA 3		
Actividades	Tareas	
MSI-1 Registro de la Petición	1.1	Registro de la Petición
	1.2	Asignación de la Petición
MSI-2 Análisis de la Petición	2.1	Verificación y Estudio de la Petición
	2.2	Estudio de la Propuesta de Solución
MSI-3 Preparación de la Implementación de la Modificación	3.1	Identificación de Elementos Afectados
	3.2	Establecimiento del Plan de Acción
	3.3	Especificación del Plan de Pruebas de Regresión
MSI-4 Seguimiento y Evaluación de los Cambios hasta la Aceptación	4.1	Seguimiento de los Cambios
	4.2	Realización de las Pruebas de Regresión
	4.3	Aprobación y Cierre de la Petición

Tabla 4-6. Actividades y tareas del mantenimiento de sistemas de información en MÉTRICA 3.

4.3.1.2. Modelo de Kajko-Mattsson para Gestión de Problemas.

Esta investigadora sueca ha propuesto un modelo conceptual con los conceptos más importantes utilizados en la gestión del PMS. En Kajko-Mattsson (1998) presenta una lista con los siguientes conceptos: identificación y descripción de un problema (con este nombre se refiere a las peticiones de modificación), unicidad de un problema, fecha y hora de ocurrencia de un problema, criticidad y urgencia de un problema, y clasificación de las causas de problemas. En Kajko-Mattsson et al (2001a) se completa el modelo anterior con una taxonomía

de las actividades de gestión de problemas, que establece la siguiente jerarquía de grupos de actividades:

Actividades de gestión de problemas

- Producto
- Proceso
 - o Análisis y control del proceso
 - o Análisis y resolución de problemas
 - Descripción e informe del problema
 - Investigación del problema
 - Identificación de la causa del problema
 - Análisis de la causa del problema
 - Diseño de la modificación
 - Decisión de la modificación
 - Implementación de la modificación
 - o Otras
- Recurso

Esta taxonomía es muy detallada, por ejemplo, en el grupo de actividades de “diseño de la modificación” se incluyen 27 tareas y subtarefas. Además, es parte integrante del modelo llamado "*Corrective Maintenance Maturity Model*" (CM3), que se presenta en la dimensión de mejora (apartado 4.3.5.2).

Kajko-Mattsson piensa que es conveniente utilizar MP especializados para cada tipo de mantenimiento, aunque todos ellos se deberían integrar en único modelo global genérico del PMS, pero que no está desarrollado. Quizás la solución a este problema se encuentra en la experiencia acumulada en el desarrollo de la metodología MANTEMA (ver apartado 4.3.1.5), que ha permitido comprobar que los diversos tipos de mantenimiento tienen algunas actividades diferentes y otras muchas comunes.

En realidad, esta propuesta restringe la gestión del PMS a tan sólo la gestión de las peticiones de modificación (en la misma línea que MÉTRICA 3, pero de forma más restringida al abordar sólo mantenimiento correctivo), mientras que todos los estándares de ciclo de vida (ISO, IEEE) dicen que el proceso de mantenimiento es bastante más amplio.

4.3.1.3. Modelo de Ciclo de Vida de Kung y Hsu.

Kung y Hsu (1998) proponen un modelo de ciclo de vida del mantenimiento con la intención de “ayudar en la planificación del proceso”. El ciclo de vida propuesto consta de cuatro etapas, que dibujan el histórico de solicitudes de modificación de una aplicación. Las etapas son:

- 1) *Introducción*: Durante esta etapa, el uso del sistema es bajo porque el número de usuarios que lo conocen es pequeño, y las demandas de mantenimiento se refieren casi exclusivamente a peticiones de soporte técnico.
- 2) *Crecimiento*: Cada vez se va incorporando al uso del sistema un mayor número de usuarios, lo que implica un crecimiento en las solicitudes de mantenimiento para corrección de errores. Esto implica un aumento en el personal necesario para ejecutar las intervenciones.
- 3) *Madurez*: El número de errores en el sistema se estabiliza, y ahora lo que los usuarios demandan son mejoras en la aplicación, por lo que crece el número de solicitudes de mantenimiento perfectivo. Durante esta etapa, los equipos de mantenimiento se dedican al aumento de las prestaciones del software y a intentar prolongar la vida del sistema.

- 4) *Declive*: En esta etapa los usuarios demandan el uso de nuevas técnicas y entornos, por lo que se tiende a disminuir las peticiones de modificación de todo tipo y a sustituir la aplicación (retirada), o bien a integrarla con otros sistemas más modernos (migración).

Estos autores representan gráficamente la distribución de peticiones de modificación comentada en las cuatro etapas anteriores con un gráfico como el mostrado en la Figura 4-1, que está basada en la experiencia en 2 proyectos reales.

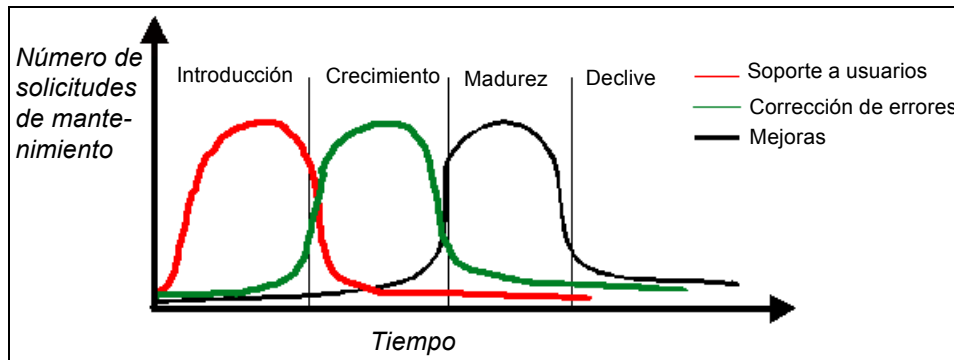


Figura 4-1. Etapas en el ciclo de vida de mantenimiento propuesto por Kung y Hsu.

En realidad, esta propuesta no es un modelo de ciclo de vida del mantenimiento (a pesar de su nombre), sino una muestra empírica de la distribución en el tiempo de las solicitudes de mantenimiento a lo largo de la vida de un producto software. En este sentido, la propuesta concuerda bastante con el modelo en etapas de ciclo de vida de un producto software planteado por Rajlich y Bennett (2000), que se comentó en la introducción. En concreto, la experiencia de Kung y Hsu confirma experimentalmente lo que dijimos en el capítulo 1: cada etapa del ciclo de vida de un producto software se caracteriza porque es mayoritario un determinado tipo de mantenimiento.

4.3.1.4. Modelo de Mantenimiento de Stoecklin, Williams y Stoecklin.

Estos tres autores proponen un enfoque de gestión del PMS que puede resumirse en el siguiente algoritmo (Stoecklin et al, 1998):

```

SI el mantenimiento es urgente ENTONCES
  REPETIR
    Analizar el problema
    Inspeccionar el código
    Modificar el código
    Realizar pruebas y entregar
  HASTA que el cambio sea satisfactorio
SI NO
  Entrevistar a expertos para determinar los requisitos de la modificación
  Definir el impacto del mantenimiento sobre los objetos de configuración del software
  Adaptar un plan inicial de mantenimiento basado en las tareas del proceso de mantenimiento
  En caso necesario, modificar el plan para planificar recursos
  Implementar la modificación utilizando el plan del proyecto
FIN SI
  
```

Este trabajo no propone un modelo para el PMS, sino una parte de lo que es la práctica habitual en las organizaciones de mantenimiento. La clave de la propuesta está en que, recibido un informe de un problema, se propone utilizar una técnica de “análisis de impacto sobre la configuración software” para determinar el plan de mantenimiento a seguir; que viene a ser parecido a decidir cuál es el tipo de mantenimiento que corresponde para ese problema y, en consecuencia, asignar un plan de mantenimiento preestablecido para cada tipo. Únicamente en el caso de mantenimiento correctivo urgente (no planificado por definición) no se podría utilizar dicha técnica.

4.3.1.5. Metodología MANTEMA.

MANTEMA es una metodología especialmente diseñada para el PMS, que ha sido elaborada por el grupo Alarcos de la Universidad de Castilla-La Mancha, al cual pertenece el autor de esta tesis, como resultado del proyecto MANTEMA (ver capítulo 2). Su versión 2 ha sido el principal resultado de la tesis doctoral de Macario Polo (Polo, 2000), también miembro del grupo Alarcos. Por esta razón, aunque la metodología MANTEMA es un componente fundamental del entorno MANTIS (de hecho, tal y como se explicará en detalle en el capítulo 4, MANTEMA es la metodología base propuesta en el entorno MANTIS para llevar a cabo el proceso de mantenimiento), no es un objetivo principal de este trabajo. Por esta razón, tan sólo se ha incluido un resumen de sus principales características en el anexo B, que se complementa con las correspondientes anotaciones en los capítulos 5 y 6, cuando se comenten aspectos concretos que se han incorporado a dicha metodología como resultado de su inclusión en el entorno MANTIS.

4.3.1.6. Otras Propuestas.

En los últimos años, la “moda” de la “programación extrema” (*eXtreme Programming*, XP) ha hecho que algunos autores propongan llevar a cabo una gestión ligera del PMS.

En esta línea, Poole et al (2001) presentan una experiencia empresarial de implementación de prácticas de XP en proyectos reales de mantenimiento de software. Estos autores comentan que, aunque tienen algunos problemas pendientes de resolver, los resultados están siendo buenos. En concreto, parece que las principales ventajas de este estilo ligero de gestionar proyectos software son, según dichos autores, la gran importancia que se concede a las pruebas y el papel más importante que juegan las métricas y, en general, el proceso de medida. La importancia de ambos aspectos en el PMS es evidente y, por tanto, no es de extrañar que los resultados sean buenos. En cualquier caso, no quedan suficientemente claros, en dicho artículo, los problemas que pueden surgir al aplicar estos métodos a procesos complejos de mantenimiento, que son la gran mayoría.

Fioravanti (2002) también ha estudiado las repercusiones de las técnicas de XP en el PMS, pero en este caso, referido a cómo se ven afectados los costes posteriores de mantenimiento en proyectos de desarrollo de software. La promesa principal de XP es mantener constantes los costes de mantenimiento; lo cual contrasta con los métodos más tradicionales caracterizados por el crecimiento casi exponencial de dichos costes (ver Figura 4-2). Este autor comenta en su trabajo la manera en que cada una de las prácticas que caracterizan la XP puede contribuir a alcanzar esta promesa. En cualquier caso, Fioravanti dice que XP no es una panacea para reducir los costes del mantenimiento, ya que la aplicación de todas las prácticas XP de

forma adecuada es bastante difícil, especialmente en proyectos industriales con grupos de más de 10 personas.

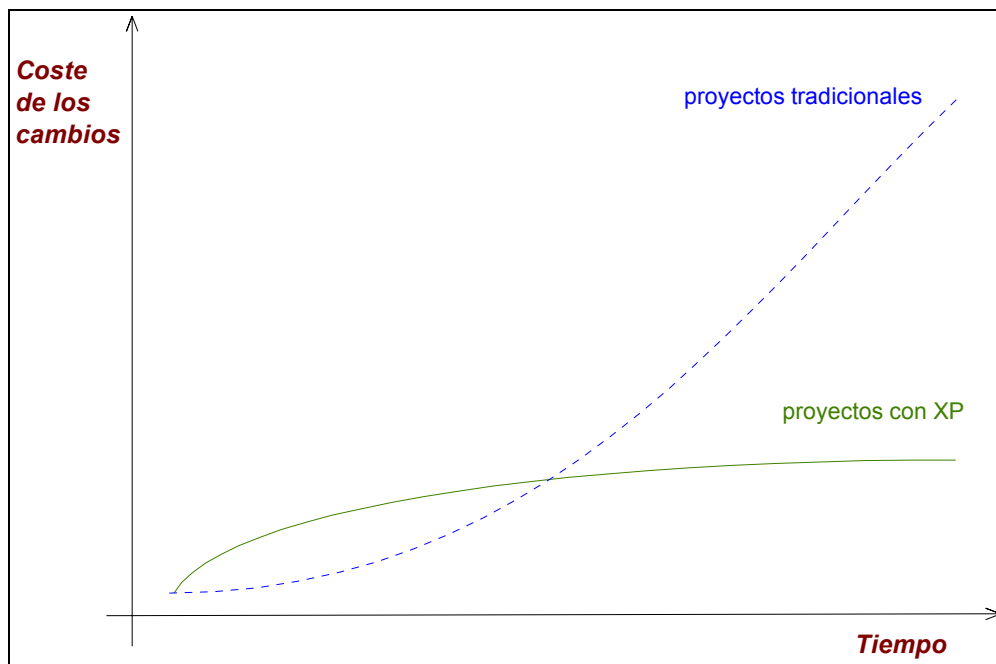


Figura 4-2. Evolución de los costes de mantenimiento en proyectos clásicos vs XP.

Otras propuestas, que plantean otros modos originales de abordar de forma global la gestión del mantenimiento del software, son:

- Un modelo arquitectural del software como un servicio (en vez de cómo un producto), orientado a conseguir la evolución rápida (Bennett et al, 2001).
- La utilización de Sistemas de Gestión de Flujos de Trabajo (*WorkFlow Management Systems*) para gestionar procesos de mantenimiento realizados por grupos virtuales de personas, es decir, cuando la asignación de roles a individuos no es fija (Aversano et al, 2001).
- El “árbol de evolución”, un modelo del ciclo de vida del software, propuesto por Tomer y Schach (2000), que define el proceso de desarrollo como una evolución continua. En esta propuesta se da completamente la vuelta a la tortilla respecto de planteamientos tradicionales que relegaban el PSM a un caso particular de desarrollo. Para estos autores, el desarrollo es un caso especial del proceso de mantenimiento, que se caracteriza por estar basado en un árbol de decisiones ingenieriles que se deben adoptar en diferentes momentos. Estas decisiones son tomadas por los ingenieros software en respuesta a cambios en los requisitos.

4.3.2. Gestión del Proyecto.

Esta dimensión de la gestión se refiere a la identificación, establecimiento, coordinación y supervisión de las actividades, tareas y recursos necesarios para culminar con éxito un proyecto (ver apartado 3.3.2). Aunque es prácticamente imposible separar estos aspectos de los generales de gestión (comentados en el apartado anterior) y de otras dimensiones de la gestión,

comentamos a continuación algunas propuestas que formulan aportaciones centradas en los aspectos citados.

4.3.2.1. Planificación.

Birk et al (1998) han propuesto una aproximación sistemática a la planificación de proyectos de mantenimiento basada en objetivos. Estos investigadores buscan planificar el PMS teniendo en mente su mejora continua. El elemento central de esta aproximación son las llamadas dependencias producto/proceso (PPD's). En la Figura 4-3 se muestra un ejemplo de PPD, que representa el impacto que las inspecciones software (una práctica de ingeniería del software) tienen en el coste (una propiedad del producto software resultante) cuando se utilizan para verificación de código durante las actividades de implementación y pruebas (según IS 15504). Las flechas discontinuas indican impactos que justifican la dependencia. En orden a su reutilización, las PPD's deben ser modeladas explícitamente (modelos de PPD's), empaquetadas con la información de contexto relevante (paquetes de experiencia de PPD's: PPD-EP's), y almacenadas en una base de experiencias.

La propuesta incluye también un método para construir la base de experiencias, que consta de dos fases con varias sub-fases:

- *Construcción* de los modelos de PPD's: formada por las sub-fases de “Entrevistas y encuestas técnicas”, “Minería de datos” y “Evaluación de productos y procesos”. En ésta última sub-fase, se propone utilizar las técnicas GQM (Basili et al, 1994) y Bootstrap (Koch, 1993) para la evaluación del producto y el proceso, respectivamente.
- *Validación* de los modelos de PPD's mediante técnicas empíricas y entrevistas.

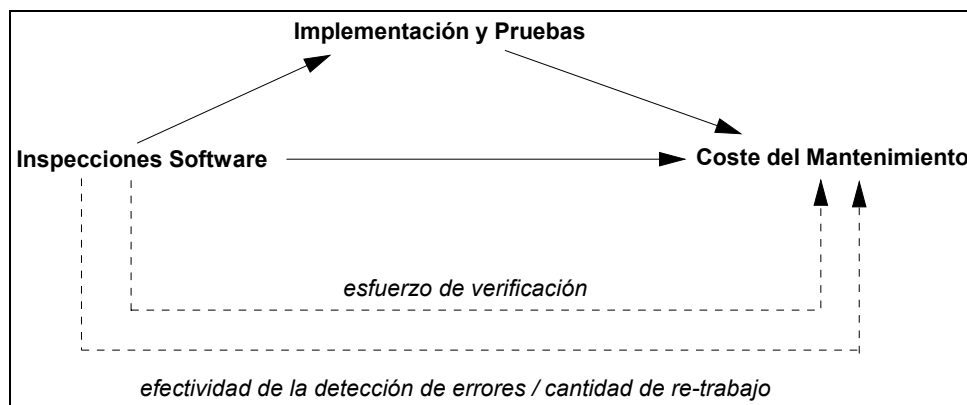


Figura 4-3. Un ejemplo de dependencia producto/proceso (PPD).

En resumen, la propuesta anterior se centra en sacar provecho de la experiencia adquirida en proyectos anteriores, sugiriendo un método para planificar proyectos de mantenimiento basado en integrar “gestión del conocimiento”. Las PPD's y los PPD-EP's son una manera original y específica de estructurar dicho conocimiento.

4.3.2.2. Estimación del Esfuerzo de Mantenimiento.

Existen muchas propuestas que utilizan métodos y métricas diversos para la estimación del esfuerzo de mantenimiento. La causa de que este asunto sea uno de los que más atención ha recibido dentro del mantenimiento del software es que la estimación del esfuerzo de las intervenciones de mantenimiento es muy importante para los gestores cuando planifican proyectos de mantenimiento y cuando se tienen que realizar análisis costes/beneficios para tomar decisiones. Quizás la propuesta más conocida sea COCOMO 2 (Boehm et al, 1995 y 2001), pero su adaptación y precisión en el caso de proyectos de mantenimiento deja bastante que desear, debido a que está orientada a proyectos de desarrollo.

En suma, la estimación del esfuerzo es uno de los temas más maduros científicamente en el ámbito del PMS. Por esta razón, en los apartados próximos nos limitaremos a presentar algunas de las últimas propuestas más novedosas. En cualquier caso, para profundizar en este asunto existe bastante bibliografía disponible, incluso en castellano. Por ejemplo, el capítulo 5 del libro de Piattini et al (2000) y el apéndice VI del informe técnico de Polo et al (1999b) están dedicados en especial a este tema.

También son interesantes el análisis comparativo de diferentes métodos de estimación realizado por Jorgensen (1995) y el estudio de Slaughter y Banker (1996) sobre los efectos del uso de diversas prácticas y herramientas de desarrollo de software en el esfuerzo posterior de mantenimiento. En este último caso, los autores han desarrollado un marco general basado en la complejidad del software que le ha permitido sacar la conclusión (y comprobar mediante un caso de uso) de que el esfuerzo de mantenimiento se reduce con buenas prácticas de desarrollo, incluyendo la programación estructurada, el uso de generadores de informes y el software empaquetado. Curiosamente, también se comprobó que el esfuerzo se incrementa cuando se utilizan generadores de código.

4.3.2.2.1. Variantes de COCOMO.

Una variante del método COCOMO ha sido propuesta por Granja y Barranco (1997), que realizan una serie de modificaciones para mejorar su aplicación al mantenimiento. Estos autores introducen un “índice de mantenibilidad”, que mide la facilidad de mantenimiento del software considerado. De esta manera, el coste de mantenimiento (CM) resultante es

$$CM = TCA * CD * IM$$

siendo TCA la tasa de cambios anual, CD el coste de desarrollo, e IM el índice de mantenibilidad. A su vez, éste índice está formado por la suma de otros tres:

$$IM = IC + IR + IP$$

siendo

IC = índice de comprensibilidad (de los cambios que deben hacerse);

IR = índice de la realización de los cambios; e

IP = índice de las pruebas de los cambios.

Estos índices se obtienen a partir de las métricas correspondientes: porcentaje de líneas comentadas por cada cien, para calcular IC; porcentaje de líneas sin datos constantes por cada cien, para calcular IR; y número de errores comprobados por cada cien líneas de código, para calcular IP.

El problema de esta propuesta es que no es útil para la estimación del esfuerzo de mantenimiento, ya que la información disponible cuando se realiza dicha estimación (durante la

planificación) no es suficiente para calcular las métricas utilizadas para obtener el índice de mantenibilidad.

4.3.2.2.2. Modelos Dinámicos.

Calzolari et al (1998) proponen un original método para modelar el esfuerzo de mantenimiento utilizando sistemas dinámicos, que está basado en el modelo “predador/presa”. La idea básica de este modelo es que, en ausencia de presas, los predadores se extinguen y, en ausencia de predadores, la población de presas alcanza y sobrepasa la capacidad del entorno para alimentarlas. En los casos intermedios se alcanza un equilibrio estable entre predadores y presas. La similitud del esfuerzo de mantenimiento con este modelo dinámico biológico se basa en que:

- a) El mantenimiento correctivo es esencialmente predador de defectos software, y el esfuerzo de mantenimiento se alimenta de errores descubiertos por el usuario.
- b) Los mantenimientos perfectivo y adaptativo se alimentan de necesidades del usuario, y el esfuerzo de estos dos tipos de mantenimiento se adapta a la cantidad de solicitudes de ambos tipos.

El método citado ha sido aplicado al caso del mantenimiento correctivo. A pesar de su originalidad, las conclusiones obtenidas por los autores no lo han sido tanto porque ya eran conocidas con anterioridad y totalmente previsibles. Estas conclusiones son:

- Se prevén aumentos de correctivo inmediatamente después de cada nueva versión de un software. Tras este incremento, el esfuerzo de mantenimiento disminuye.
- La duración del esfuerzo de mantenimiento dependerá del grado en que se haya modificado el producto software.

4.3.2.2.3. Puntos-Función.

El estudio más conocido sobre estimación del esfuerzo de mantenimiento utilizando puntos-función (PF) es el realizado por Niessink y Van Vliet (1997). Estos investigadores realizan ciertos análisis comparativos entre el esfuerzo de mantenimiento de varias intervenciones y el número de PF modificados, contados según métodos diferentes. Los resultados obtenidos no son satisfactorios para ningún modelo predictivo, pero sí que se observa que el esfuerzo de mantenimiento es mucho más dependiente del tamaño del componente que se va a cambiar, que del tamaño del propio cambio (medidos ambos parámetros en PF). Es decir, que

$\text{Esfuerzo} \approx K \times \text{Tamaño del componente} \times (1 + \delta \times \text{Tamaño del cambio})$
en vez de

$\text{Esfuerzo} \approx K \times \text{Tamaño del cambio}$

Esta falta de idoneidad de los PF para realizar estimaciones ha sido también discutida en Dolado y Fernández (1999). Así, por ejemplo, Niessink y Van Vliet indican que no son aplicables a la mayoría de las intervenciones de mantenimiento correctivo ya que éstas afectan, por lo general, a sólo unas pocas líneas de código que no llegan a constituir, ni siquiera, un PF.

A pesar de todo, son muchas las organizaciones que utilizan PF para estimar el coste de las intervenciones de mantenimiento, especialmente para las de tipo perfecto.

4.3.2.2.4. Gestión del Conocimiento.

Bengtsson y Bosch (1999) proponen un método para predecir (estimar) el esfuerzo de mantenimiento a partir de la siguiente información: especificación de requisitos, diseño arquitectural, experiencia de los ingenieros de mantenimiento y, si están disponibles, datos históricos. Para concretar los requisitos y analizar la arquitectura se utilizan “escenarios”. El método consta de las seis etapas siguientes:

1. Identificar categorías de tareas de mantenimiento.
2. Síntesis de escenarios (un conjunto concreto para cada categoría de tareas).
3. Asignar a cada escenario un peso (indicador de la probabilidad de que dicho escenario tenga cambios).
4. Estimar el tamaño de todos los elementos software.
5. Estimar el impacto de la realización de cada escenario (en la arquitectura y en los elementos software).
6. Calcular el esfuerzo de mantenimiento (haciendo la media ponderada de los valores de lo estimado en la etapa 5 con los pesos asignados en la etapa 3).

La principal aportación de esta propuesta no es ninguna técnica concreta, sino la idea genérica de hacer la estimación mediante un método organizado y basado en aplicar escenarios diferentes para cada tipo de tarea. Para profundizar en técnicas concretas, existen bastantes publicaciones. Por ejemplo, sobre cómo realizar el análisis de impacto del paso 5, Sneed (2001) propone un método aplicable a sistemas distribuidos orientados a objetos; y O’Neal y Carver (2001) sugieren otro método basado en la trazabilidad de ciertos atributos de los requisitos.

4.3.2.3. Control de la Productividad.

Entre las propuestas que abordan problemas de supervisión y seguimiento de proyectos de mantenimiento, quizás uno de los problemas principales estudiado es el control de la productividad. En este tema, Chan (2000) plantea que, desde el punto de vista del interés de los usuarios del mantenimiento, las maneras tradicionales de medir la productividad no son útiles. Aunque habitualmente la productividad del mantenimiento es medida por el número de horas consumido en atender una petición de modificación, Chan dice que esta medida captura el coste para el departamento responsable del mantenimiento, pero ignora los costes potenciales que repercuten en los usuarios debidos a los tiempos de espera hasta que su demanda es completamente satisfecha. Por esta razón, este investigador propone utilizar el tiempo de espera de los usuarios como medida de la productividad del mantenimiento. Chan define dicho tiempo de espera como la suma del tiempo que la petición pasa en la cola de peticiones y el tiempo que dura su realización.

4.3.3. Gestión de la Calidad.

En esta dimensión de gestión el interés se centra en la calidad de los productos y procesos, a nivel de proyecto y a nivel de la organización en su conjunto. Las técnicas concretas de aseguramiento de calidad no están incluidas en este documento porque pertenecen al proceso de soporte correspondiente (apartado 3.3.2).

El aumento de los recursos humanos y económicos dedicados al PMS puede suponer una solución a corto plazo, pero para resolver el problema a largo plazo se hace necesario adoptar una aproximación que permita mejorar la calidad del proceso en su conjunto. Los métodos para aumentar la calidad del proceso de mantenimiento se parecen cada vez más a los empleados en la industria en general (Piattini et al, 2000). Por ejemplo, pueden utilizarse métricas de producto y de proceso, o emplear mejores herramientas CASE, especialmente si están integradas en un EIS que facilita su interoperabilidad.

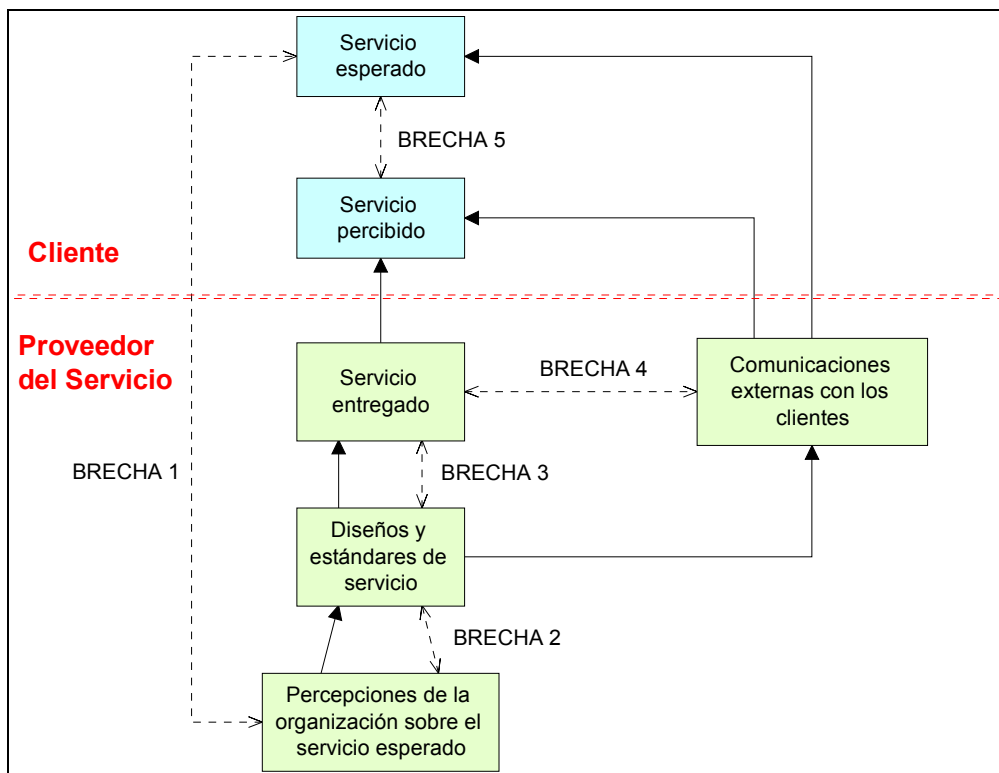


Figura 4-4. Modelo de "brechas" de calidad de servicio.

4.3.3.1. Calidad de Servicio de Niessink y Van Vliet.

Los servicios se distinguen de los productos en que son intangibles, heterogéneos, perecederos y su producción y consumo son simultáneos. Esta distinta naturaleza de los servicios y los productos influye en la manera en que los clientes evalúan sus respectivas calidades. En particular, la calidad de un servicio viene determinada por dos factores: la calidad técnica (cuál es el resultado del servicio) y la calidad funcional (la forma en que el cliente recibe el servicio).

Niessink y Van Vliet (2000), se basan en el razonamiento anterior para justificar que el mantenimiento de software, al contrario que el desarrollo, debe ser visto más como la provisión

de un servicio que como la fabricación de un producto. En consecuencia, estos autores prevén que los clientes juzgarán de forma diferente la calidad de los proyectos de mantenimiento y la calidad de los proyectos de desarrollo de software y; por tanto, para lograr resultados de alta calidad en mantenimiento de software se deberán llevar a cabo acciones distintas que en desarrollo.

Estos autores analizan la calidad del mantenimiento de software basándose en el modelo de “brechas” de calidad de servicio (Parasuraman et al, 1985). En este modelo (ver Figura 4-4), la diferencia (brecha) entre la calidad percibida de los servicios y la calidad esperada (brecha 5) está causada por otros 4 tipos de brechas diferentes (brechas 1 a 4).

Para que las organizaciones ofrezcan servicios de mantenimiento exitosos, es necesario que reduzcan la brecha 5, que es la importante desde el punto de vista de los clientes. Para lograr esta reducción, hace falta atajar las cuatro brechas que son la causa original. Con esta justificación, Niessink y Van Vliet piensan que las organizaciones de mantenimiento exitosas (con calidad de servicio) se diferencian de las que no lo son en que realizan las siguientes actividades:

- Gestión de compromisos para reducir la brecha 1. Los compromisos o acuerdos de mantenimiento deben ser planificados y documentados. Este trabajo será mejor si el mantenedor y el cliente elaboran juntos un “contrato de nivel de servicio” (CNS) basado en las necesidades del cliente.
- Planificación del mantenimiento para reducir la brecha 2. Las actividades de mantenimiento especificadas en el CNS tienen que ser planificadas. Esta planificación incluye a las actividades de mantenimiento propiamente dichas, la transferencia de resultados al cliente, la estimación de los recursos necesarios, la elaboración de un calendario, y la identificación de los posibles riesgos.
- Seguimiento de las actividades de mantenimiento para reducir la brecha 3. El propósito de este trabajo es proveer información sobre la realización de las actividades de mantenimiento, y tomar acciones correctivas si fuese necesario para garantizar el cumplimiento del CNS.
- Gestión de eventos para reducir la brecha 4. Esta actividad se refiere al manejo organizado de los eventos que pueden causar desviaciones respecto de los niveles de servicio acordados. Estos eventos pueden ser peticiones de cambio en el software por parte de los usuarios u otros participantes, o incidentes imprevistos.

4.3.3.2. Otras Propuestas.

Schneidewind (1999a y 2000) ha elaborado un modelo de control de calidad y predicción para el PMS. Este modelo permite identificar los módulos que requieren atención prioritaria y permite hacer una predicción (estimación) de la calidad del mantenimiento. La principal aportación de esta propuesta no es un modelo de calidad del proceso de mantenimiento, ni siquiera un modelo de calidad en general al estilo de la norma ISO 9126 (ISO/IEC, 2001c), sino un marco formal para la utilización de métricas de calidad. Realmente, la originalidad de la propuesta está en la técnica de análisis para decidir cuales y cuantas métricas de calidad utilizar. Esta técnica está basada en medir la contribución marginal de cada métrica en la satisfacción de unos determinados criterios estadísticos y de utilidad propuestos por el autor; y en la agrupación de las métricas en funciones discriminantes booleanas, que son un nuevo tipo de discriminante para clasificar atributos de calidad del software.

En cuanto a la evaluación de la calidad del proceso de mantenimiento, la única bibliografía conocida es la de Visaggio (1999), que expone los resultados de un experimento controlado, cuyo objetivo es evaluar y comparar algunas características de calidad para los paradigmas de mantenimiento propuestos por Basili (1990). Estos paradigmas (diferentes modelos de proceso genéricos para realizar el mantenimiento) son:

- *Reparación Rápida (Quick Fix, QF)*, que consiste en hacer los cambios necesarios en el código lo antes posible y después retocar la documentación.
- *Mejora Iterativa (Iterative Enhancement, IE)*, que consiste en modificar los documentos de más alto nivel afectados por los cambios y, entonces, propagar dichos cambios hacia abajo hasta llegar al código.
- *Reutilización Total (Full Reuse, FR)*, que consiste en construir un sistema nuevo utilizando los componentes del sistema antiguo y otros que estén disponibles.

Este estudio comparó los paradigmas QF e IE realizando dos ocurrencias del mismo experimento, una con estudiantes de últimos años de carrera y otra con profesionales experimentados. No se evaluó el paradigma FR porque no se disponía de la tecnología adecuada para gestionar el repositorio de componentes software y poder reutilizarlos. De la lectura del artículo se concluye que, en realidad, el objetivo real y concreto del experimento fue analizar las características de dichos paradigmas en cuanto a su relación con el nivel de comprensibilidad del software resultante desde el punto de vista del mantenedor. Las principales conclusiones obtenidas en el experimento son:

- QF es más costoso en tiempo que IE, pero no de manera determinante.
- QF daña la comprensibilidad del sistema más que IE: sus valores de corrección, completitud y trazabilidad resultantes son más bajos.
- En cualquier caso, el mantenimiento pone en peligro la calidad del software ya que, ni siquiera IE está libre de efectos perjudiciales.

4.3.4. Gestión de Riesgos.

La diferente naturaleza del PMS frente al proceso de desarrollo hace que los riesgos en los proyectos de mantenimiento puedan ser diferentes y con distinta importancia a los habituales en proyectos de desarrollo.

Es importante notar que, aunque hay buenos trabajos que analizan la problemática de la identificación, estimación y gestión de riesgos una vez el proceso de mantenimiento ha comenzado (como los dos analizados en esta sección), no existen guías que ayuden en la identificación y estimación de riesgos durante las etapas iniciales del mantenimiento. Las técnicas de estimación de costes constituyen quizás una excepción en este punto; pero no se consideran métodos de estimación de riesgos ni incluyen factores de riesgos en sus estimaciones o, si lo hacen, éstos se confían al juicio de un experto sin proporcionar más detalles (Briand et al, 1998a).

Por otro lado, las metodologías conocidas de gestión de riesgos en sistemas de información, en general, no dedican una atención adecuada a los riesgos del PMS. Este es el caso de MAGERIT (*Metodología de Análisis y GEstión de Riesgos de los sistemas de Información de las AdminisTraciones Públicas*), elaborada por el Ministerio de Administraciones Públicas de España (CSI, 2002). Por ello, sería necesario llevar a cabo más

estudios específicos sobre riesgos del PMS. De todas maneras, algunas de las propuestas generales de gestión de riesgos en ingeniería del software son utilizables en mantenimiento, al menos, eso es lo que afirma, por ejemplo, Kontio (2001) en su tesis doctoral dedicada a este asunto, en referencia al método RisKit, que el mismo autor ha desarrollado.

4.3.4.1. Criterios y Factores de Riesgo de Schneidewind.

Schneidewind (1997) propone una serie de criterios y métricas para estimar los riesgos del mantenimiento de sistemas críticos (por ejemplo, misiones espaciales); basándose, principalmente, en el tiempo total de prueba del sistema, el número de errores encontrados durante las pruebas, y el tiempo transcurrido entre cada par de errores. Con estos datos, el autor propone una ecuación para calcular el instante en que ocurrirá el siguiente fallo, de manera que el proyecto pueda realizarse (la misión espacial pueda despegar) si su finalización tendrá lugar antes de que ocurra el próximo fallo crítico. Además, estos valores pueden utilizarse también como parámetros para la medición de la calidad del proceso de mantenimiento (ver apartado 4.3.3); de tal manera, que un incremento en el número de errores por unidad de tiempo aconseja revisar el PMS que se está siguiendo. Al igual que se comentó en el apartado de calidad, las aportaciones más interesantes de este trabajo están en el ámbito de las métricas.

Trabajos posteriores del mismo autor han abordado el problema de identificar los factores de riesgo debidos a cambios en los requisitos y su utilidad como predictores de problemas en la fiabilidad y mantenibilidad del software (Schneidewind, 2001 y 2002). Puesto que existe una relación indirecta entre estos conceptos (los cambios en los requisitos pueden producir incrementos en el tamaño y complejidad del software, lo que, a su vez, puede afectar negativamente a su fiabilidad y mantenibilidad), este autor utiliza métricas de tamaño y complejidad para cuantificar dicha relación. El estudio empírico correspondiente ha sido realizado sobre una colección de peticiones de cambio en el software que controla una lanzadera espacial (*Space Shuttle*). Este caso es de especial interés debido a la alta criticidad que este producto tiene.

Como resultado, Schneidewind define 19 factores de riesgo que agrupa en varias categorías. En la Tabla 4-7 se indican dichos factores de riesgo. Para cada uno, se muestra su nombre, descripción, la pregunta que ayuda a determinar si existe o no un riesgo alto con respecto a dicho factor, y si se ha detectado una relación estadística entre el factor y la fiabilidad y mantenibilidad del software.

Factores de riesgo de los cambios en los requisitos				
Categoría	Factor	Descripción	Pregunta	RE ¹³
Complejidad	Complejidad	Evaluación cualitativa de la complejidad del cambio	¿Es este cambio altamente complejo en relación con otros cambios hechos anteriormente?	No sign
	Mods	Número de modificaciones o iteraciones en el cambio propuesto	¿Cuántas veces debe el cambio ser modificado o presentado al Responsable de Control de Cambios (RCC) antes de su aprobación?	Sign

¹³ Relación Estadística encontrada entre el factor de riesgo y la fiabilidad y mantenibilidad del software: Sign => Significativa; No sign => No significativa; Dat Ins => Datos Insuficientes.

Factores de riesgo de los cambios en los requisitos				
Categoría	Factor	Descripción	Pregunta	RE ¹³
<i>Tamaño</i>	Sloc	Número de líneas de código fuente afectadas por el cambio	¿Cuántas líneas de código deben ser modificadas para implementar la petición de cambio?	Sign
	Mod Chg	Número de módulos cambiados	¿Es excesivo el número de módulos cambiados?	No sign
<i>Criticidad de los cambios</i>	Crit Func	Criticidad de la funcionalidad añadida o cambiada por la petición de cambio	¿Es la funcionalidad añadida o cambiada crítica para el éxito del proyecto?	Dat Ins
	Off nom path	El cambio está en un camino de ejecución nominal o no-nominal (condición de excepción)	¿Afectará a la fiabilidad del software un cambio en un camino de ejecución no-nominal?	Dat Ins
<i>Localidad de los cambios</i>	Critic area	Area del programa afectada	¿Afectará el cambio a un área del código que es crítica para el éxito del proyecto?	Dat Ins
	Recent chgs	Cambios recientes en el código dentro del area afectada por la petición de cambio	¿Los sucesivos cambios en un área del código conducen a código poco mantenible?	Dat Ins
	New/exist code	Código nuevo o existente que está afectado	¿Un cambio que implica código nuevo genera código poco mantenible?	Dat Ins
	Fails ex code	Número de fallos del sistema que podrían ocurrir antes de que el código que implementa los requisitos sea ejecutado	¿Estarán los cambios en un camino de ejecución en el cual sólo podrían ocurrir un pequeño número de fallos del sistema antes de que el código cambiado sea ejecutado?	Dat Ins
<i>Problemas y funcionalidades de los requisitos</i>	Other chgs	Número y tipos de otros requisitos afectados por el cambio dado en los requisitos	¿Existen otros requisitos que están siendo afectados por este cambio?	Dat Ins
	Issues	Número de conflictos posibles entre los requisitos	¿Este cambio entra en conflicto con otros cambios en los requisitos?	Sign
	Prin funcns	Número de funciones principales del software afectadas por el cambio	¿Cuántas funciones importantes del software tendrán que ser cambiadas para implementar el cambio?	No sign
<i>Rendimiento</i>	Space	Cantidad de espacio de memoria requerido para implementar el cambio	¿Afectará el cambio el uso de memoria de tal manera que otras funciones no tendrán suficiente memoria para operar eficientemente?	Sign
	Cpu	Efecto sobre el rendimiento de la CPU	¿Afectará el cambio los ciclos de CPU de tal manera que otras funciones no tendrán suficiente capacidad de CPU para operar eficientemente?	Dat Ins
<i>Recursos Humanos</i>	Inspects	Número de inspecciones requeridas para aprobar el cambio	¿El número de inspecciones en los requisitos conduce a un consumo excesivo de recursos humanos?	No sign
	Manpower	Mano de obra requerida para implementar el cambio	¿La mano de obra necesaria para implementar el cambio es significativa?	No sign
	Cost	Mano de obra requerida para verificar y validar la corrección del cambio	¿La mano de obra necesaria para verificar y validar el cambio es significativa?	No sign
	Tests	Número de pruebas requeridas para verificar y validar la corrección del cambio	¿El número de pruebas necesarias para verificar y validar el cambio es significativa?	No sign

Tabla 4-7. Factores de riesgo de los cambios en los requisitos que afectan a la mantenibilidad.

Una utilidad importante de estos resultados es que los factores de riesgo que son estadísticamente significativos pueden ser utilizados para la toma de decisiones relacionadas con los riesgos de hacer cambios en los requisitos.

4.3.4.2. Análisis de Riesgos de Sherer.

Sherer (1997) opina que la gestión del PMS puede ser mejorada mediante la consideración de forma explícita del impacto del esfuerzo de mantenimiento sobre los riesgos en el proyecto, la mantenibilidad del software y su usabilidad. Con este fin, esta autora propone un método para medir el efecto del mantenimiento sobre los riesgos de fallo del software, y propone utilizar dicha información para guiar las decisiones de gestión del mantenimiento más importantes: cuando volver a desarrollar un software, priorización de los esfuerzos de mantenimiento, asignación de recursos, o planificación del calendario de versiones.

Para Sherer (1997) existen tres tipos de riesgos en el proceso de mantenimiento:

- De *proyecto*: el proyecto de mantenimiento no puede completarse debido a la falta de capacidad del personal, el software no puede mantenerse en tiempo y presupuesto, o la organización no tiene un PMS efectivo.
- De *usabilidad*: los sistemas mantenidos producirán problemas al ser utilizados. Este tipo de riesgos incluye cuatro subtipos: de funcionalidad, de fallos del software, de rendimiento, y financieros.
- De *mantenibilidad*: el sistema será difícil de mantener en el futuro, una vez que los cambios hayan sido realizados.

Esta investigadora propone una metodología para estimar el riesgo de fallo del software y la magnitud de la pérdida provocada por este fallo. Las cuatro etapas de dicho método son:

1. *Evaluación del riesgo externo*: Estudiar el entorno en que el software se ejecutará; con el objetivo de anticiparse a las posibles situaciones negativas que pueden provocar pérdidas.
2. *Evaluación de la exposición de módulos*: Estimar la magnitud de la pérdida debida a errores en un módulo. Existe una pérdida potencial en un módulo cuando éste está relacionado con alguna de las posibles situaciones negativas identificadas en la etapa anterior. La magnitud de la pérdida producida por cada módulo se obtiene multiplicando la probabilidad de uso del módulo por la consecuencia esperada de todas las situaciones negativas que pueden resultar de ese uso. Esta consecuencia esperada es la suma ponderada (según su probabilidad de ocurrencia) de las pérdidas (en dinero o en tiempo) de todas las situaciones.
3. *Estimación de la probabilidad de fallo*. Calcular el número esperado de fallos producidos por un módulo durante un periodo de tiempo debido a errores en dicho módulo. Se estima con un modelo de fiabilidad del software dependiente del tiempo.
4. *Evaluación del riesgo de fallo de módulos*. Que se calcula como el producto de la exposición del módulo (obtenido en la etapa 2) y el número esperado de fallos.

Por último, Sherer también explica cómo utilizar los resultados de la estimación de riesgos anterior para mejorar la gestión del PMS. Para ello propone un modelo de costes/beneficios que se puede aplicar durante la planificación como base para la minimización de los riesgos.

Una utilidad importante de esta propuesta es que puede usarse para estimar los riesgos de intervenciones de mantenimiento futuras y, especialmente, para conocer las partes más arriesgadas del software mantenido, permitiendo así la priorización de recursos y el análisis de diferentes alternativas.

4.3.5. Mejora.

La mejora de procesos de ingeniería del software ha recibido mucha atención, tanto por la comunidad investigadora, como por las empresas. Como principal resultado de esta importante actividad están los modelos y estándares de mejora y evaluación de procesos: IS 15504 (ISO/IEC, 1998c) y CMM (Paulk et al, 1993). En el ámbito del PMS las principales propuestas conocidas se presentan a continuación.

4.3.5.1. Perspectiva de Servicio de Niessink.

En línea con lo comentado en el apartado 4.3.3.1, Niessink (2000; 2001) considera el mantenimiento de software como un servicio IT (*Information Technologies*). Esta clase de servicios se caracteriza porque son provistos por un proveedor de servicios IT a un cliente para gestionar, mantener u operar el software y/o hardware utilizado por dicho cliente.

Este autor propone una aproximación a la mejora de servicios IT en general, aunque la aplica en particular al servicio de mantenimiento de software, es decir, a la mejora del PMS. Esta aproximación se realiza desde dos perspectivas de mejora diferentes y complementarias.

4.3.5.1.1. Mejora Basada en la Medida.

Desde esta perspectiva, la medida es usada para posibilitar actividades de mejora. El paradigma GQM (*Goal/Question/Metric*), desarrollado por Basili y Rombach (Basili et al, 1994), es propuesto para trasladar los objetivos de mejora a métricas cuyos valores se obtendrán durante el proceso de medida.

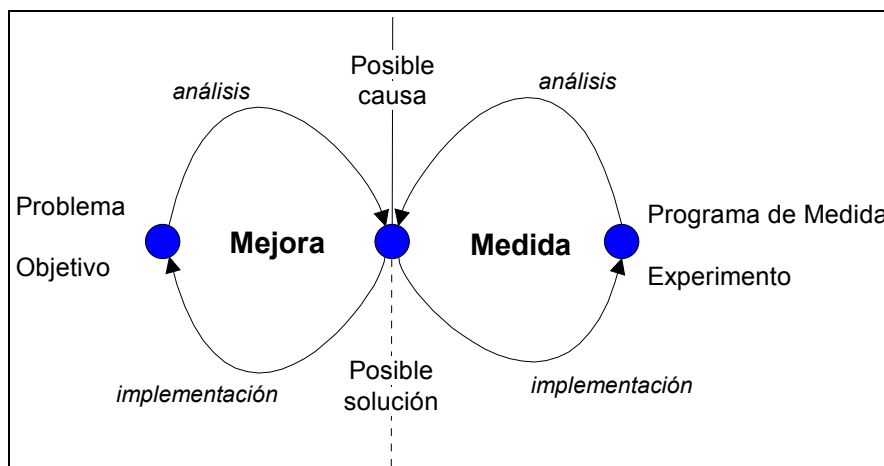


Figura 4-5. Modelo de proceso genérico para la mejora basada en la medida.

Este tipo de mejora se fundamenta en un MP genérico que se muestra resumido en la Figura 4-5. A partir del MP anterior, Niessink y Van Vliet (1998) han desarrollado un modelo de madurez de procesos de medida del software (*Measurement Capability Maturity Model*), cuyo objetivo es ayudar a las organizaciones software a estimar la madurez de su proceso de medida y les provee orientación para la mejora de dicho proceso. Este modelo es una especie de CMM especializado en el proceso de medida, con los siguientes niveles de madurez y áreas de proceso claves (KPA's):

- *Inicial*: No hay KPA's.
- *Repetible*: Diseño del proceso de medida, Colección de medidas, Análisis de medidas, Realimentación del proceso de medida.
- *Definido*: Proceso de medida organizacional, Diseño organizacional del proceso de medida, Base de datos organizacional de medidas, Programa de formación.
- *Gestionado*: Gestión de costes del proceso de medida, Selección de tecnología.
- *Optimizado*: Gestión de los cambios en el proceso de medida.

4.3.5.1.2. Mejora Basada en la Madurez.

En el apartado 4.3.3.1 ya se ha argumentado porqué, para obtener resultados de calidad, una organización que hace mantenimiento de software necesita llevar a cabo procesos diferentes que una organización que desarrolla software. Continuando este razonamiento, Niessink señala que, si se considera el PMS desde una perspectiva de servicio, los procesos de mejora necesarios para aumentar la madurez de las organizaciones de mantenimiento serán diferentes que los indicados en CMM (o en ISO 15504), ya que este modelo de madurez está concebido para procesos de producción de software. Para evitar la disfunción anterior, este investigador propone el modelo de madurez ITS-CMM (*IT Service Capability Maturity Model*) cuyos niveles de madurez y KPA's están resumidos en la Tabla 4-8.

IT Services Capability Maturity Model			
Categorías de Procesos ⇒	Gestión <i>Planificación de servicios, gestión, etc.</i>	Soporte <i>Soporte y estandarización</i>	Entrega <i>Entrega de servicio actual</i>
Niveles ↓			
Optimizado	Gestión de cambios en el proceso.	Gestión de cambios en la tecnología.	Prevención de problemas.
Gestionado	Gestión cuantitativa del servicio		Gestión de la calidad de servicio.
Definido	Gestión integrada del servicio.	Proceso Organizacional. Definición organizacional del proceso. Programa de formación.	Entrega de servicio.
Repetible	Gestión de contratos de servicio. Planificación de entrega de servicios. Seguimiento y vigilancia de servicios. Gestión de subcontratos.	Gestión de configuración. Gestión de eventos. Aseguramiento de la calidad de servicio.	
Inicial	Procesos ad hoc		

Tabla 4-8. Niveles de madurez y áreas de proceso claves en ITS-CMM.

4.3.5.1.3. Comentarios.

Incluimos esta apartado porque para analizar mejor la propuesta anterior es conveniente tener en cuenta la crítica realizada por Chapin (2000). Los principales comentarios negativos (obviamos los positivos) de este investigador son:

- Bastantes de las ideas aportadas son conocidas desde hace tiempo por los especialistas en teoría y práctica del control en organizaciones.
- Sorprende que planteando el mantenimiento de software desde una perspectiva de servicio, más en línea con la atención a los clientes, no se haya considerado ninguna referencia o publicación sobre CRM (*Customer Relations Management*).
- La definición de “capacidad de proceso de medida” es demasiado ambigua.
- En general, los aspectos humanos del problema han sido pasados por alto. Esta es, probablemente, la crítica más fundamentada ya que, como señala Chapin, “hacer mejoras significa introducir cambios en las organizaciones, es decir, es un proceso social, y no intelectual, porque habitualmente los cambios organizacionales y en procesos implican que algunas personas ganen y otras pierdan”.

A pesar de estos comentarios, la opinión del autor de esta tesis es que las ideas y aportaciones de Niessink y Van Vliet son importantes para la gestión del PMS. Aunque estas ideas están basadas en propuestas antiguas en otros campos de conocimiento, han sabido conceptualizarlas de forma adecuada para aprovecharlas por las organizaciones de mantenimiento. Además, también realizan aportaciones originales interesantes para las cuales han demostrado su utilidad con múltiples casos de uso. Buen ejemplo de ellas son el modelo genérico de mejora basada en la medida (Figura 4-5), y los dos modelos de madurez definidos: Measurement-CMM e ITS-CMM.

4.3.5.2. Modelo de Madurez para el Mantenimiento Correctivo.

Tal como su nombre indica, la propuesta “*Corrective Maintenance Maturity Model*” (CM3) está pensada sólo para mantenimiento de tipo correctivo (Kajko-Mattson, 2001). A pesar de que la autora dice que es un modelo de proceso, realmente es un modelo de madurez al estilo de CMM, cuyo sistema de procesos incluye los siguientes procesos:

- *Preentrega*: actividades para preparar el software para el futuro mantenimiento correctivo.
- *Transición*: actividades requeridas para una transferencia suave del software desde desarrollo a mantenimiento, o desde la organización de mantenimiento al cliente.
- *Gestión de problemas*: actividades necesarias para informar, analizar y resolver problemas del software (peticiones de modificación).
- *Pruebas*: actividades para probar la corrección de los cambios en el software.
- *Documentación*: actividades necesarias para crear y mantener la documentación del software y de sus cambios.
- *Interfaz de mantenimiento*: comunicar con los clientes sobre los problemas del software.
- *Formación de mantenedores*: educación y entrenamiento continuo de los ingenieros de mantenimiento. Este proceso está explicado en más detalle en (Kajko-Mattson et al, 2001b).

La propuesta CM3 es bastante amplia ya que, además de incluir un documento con las definiciones de mantenimiento y mantenimiento correctivo bastante elaboradas, explicita los siguientes componentes para cada uno de los procesos anteriores:

- Taxonomía detallada de actividades.
- Modelo conceptual (conceptos y sus interrelaciones expresados en entidad-interrelación extendido).
- Elementos de mantenimiento (objetivos y motivaciones de las actividades).
- Roles.
- Fases del proceso.
- Niveles de madurez del proceso.
- Guía del proceso (ayuda).

Adicionalmente, en CM3 se establecen tres fases principales en el ciclo de vida del PMS (pre-entrega, transición y post-entrega) y tres niveles de madurez (inicial, definido y optimizado). Dichos niveles están basados en la siguiente definición:

“Madurez del proceso de mantenimiento correctivo es la madurez y prontitud de la organización para definir, realizar, gestionar, medir, y controlar su proceso de mantenimiento correctivo. La madurez refleja la habilidad de la organización para: resolver los problemas del software, aplicar el proceso de mantenimiento correctivo de forma consistente dentro de la organización, y utilizar el proceso de mantenimiento correctivo como un vehículo de aprendizaje para mejorar los procesos de desarrollo y mantenimiento y para prevenir futuros problemas”.

En esta propuesta existe un problema de nomenclatura ya que se utilizan nombres, como proceso o actividad, para conceptos diferentes y provoca confusión al lector. Quizás sea esta la razón por la que no está clara la necesidad de establecer fases de un ciclo de vida del PMS, pues se pierde la perspectiva general necesaria: un ciclo de vida (el del software), un proceso (el de mantenimiento), y una serie de actividades (que en realidad serían lo que Kajko-Mattson denomina procesos). Otra posible causa de establecer dichas fases del mantenimiento es que la autora piense que los tipos de mantenimiento que se deben realizar difieren a lo largo de la vida de un producto software (a pesar de que CM3 se centra exclusivamente en correctivo), en cuyo caso serían aplicables los mismos comentarios que se hicieron al modelo de ciclo de vida de mantenimiento de Kung y Hsu (apartado 4.3.1.3).

4.3.5.3. Otras Propuestas.

Entre las propuestas de mejora, relacionadas con la evaluación de los procesos y organizaciones de mantenimiento, cabe citar la formulada por Briand et al (1998b). Estos autores proponen el método inductivo Q-MOPP (*Qualitative Evaluation of Maintenance Organizations, Processes and Products*) para la evaluación cualitativa del PMS, y en consecuencia, identificar sus problemas y necesidades. Q-MOPP consta de dos fases: modelado descriptivo y análisis. El resultado de la primera fase es una descripción comprensible de qué es lo que está ocurriendo durante el mantenimiento de una nueva versión del software (en sus dimensiones de organización, proceso y producto). La segunda fase utiliza este modelo descriptivo para identificar áreas que necesitan mejoras y proveer propuestas para conseguirlas.

En el artículo citado se detallan las técnicas concretas que se sugieren para llevar a cabo dichas fases, así como un caso de estudio completo de un proyecto aeroespacial de la NASA. Lo más interesante del artículo son, probablemente, las 14 lecciones aprendidas que los autores sacan de los proyectos reales en que utilizaron el método.

Por otro lado, Hall et al (2001) han realizado un estudio de los problemas que un grupo seleccionado de empresas británicas tienen al realizar proyectos de mantenimiento de software. La conclusión más significativa que sacan es que “el software está siendo entregado en el tiempo acordado pero sabiendo que tiene fallos, dando prioridad al tiempo frente a la calidad”.

4.3.6. Gestión de Recursos Humanos.

El recurso fundamental y clave para el mantenimiento del software es el humano. Por tanto, una manera de mejorar el PMS podría ser constituir un grupo separado de ingenieros de mantenimiento. Sin embargo, debido al carácter poco atractivo de este trabajo, es habitual que el personal nuevo recién incorporado sea asignado a esta actividad. En estos casos, dichos programadores inexpertos deben intentar comprender la lógica de diseño del sistema, a pesar de que no pueden comprender el modelo conceptual del software debido a que carecen de experiencia de uso de las técnicas de ingeniería del software y de conocimiento del dominio de lo que el programa realiza. Así, raramente saben cómo encontrar y corregir defectos o realizar modificaciones. Todo esto conduce a que los ingenieros de mantenimiento suelen tener una opinión negativa de su trabajo (Tan y Gable, 1998).

Por otro lado, puesto que las tareas relacionadas con el PMS comienzan mucho antes de que se realice la primera petición de modificación, es muy aconsejable que se establezca una organización del equipo de mantenimiento, estableciendo claramente las personas que participarán en cada actividad para tratar de evitar que el proceso se realice “como se pueda”. Esta organización puede ser creada formalmente o simplemente constituirse de hecho, pero, en cualquier caso, se deberán establecer claramente los procedimientos de evaluación, control, supervisión e información de cada petición de modificación. Existen muchas alternativas sobre cómo organizar el equipo de mantenimiento, aunque es esencial, incluso en pequeños equipos, establecer una delegación de responsabilidades. Entre las propuestas que abordan estos problemas, se encuentran las siguientes:

- La adopción de la teoría de colas para evaluar la dotación de personal, la gestión del proceso y la evaluación del nivel de servicio en proyectos de mantenimiento a gran escala en una factoría software virtual (Antoniol et al, 2001). La teoría de colas ayudó a la toma de decisiones de distribución de recursos humanos para equilibrar el trabajo de mantenimiento entre diversos centros que colaboraban en el proyecto.
- Un método para dimensionar la plantilla de personal de proyectos de mantenimiento críticos, que está basado en la estimación del número de peticiones de modificación que se tendrán y en cuantificar el nivel de preparación del equipo de personas disponibles (Ramaswamy, 2000). El autor utiliza la teoría de colas para realizar ambas estimaciones.
- Aplicar las ideas del campo de la medicina conocido como “procesamiento humano de información” (*Human Information Processing*, HIP) para estimar la productividad de individuos, equipos y organizaciones de mantenimiento (Ramanujan et al, 2000). Para poder aplicar dichas ideas, los autores han elaborado un modelo teórico propio de esfuerzo de mantenimiento.

4.3.7. Infraestructura.

En esta dimensión de gestión se engloban los aspectos relacionados con el uso de herramientas y EIS para gestionar el PMS. No se conoce ningún trabajo, ajeno a los desarrollados por el grupo Alarcos de la UCLM, sobre EIS para gestión del mantenimiento. En cambio, sí existen diversos trabajos sobre el uso de herramientas software para gestionar el mantenimiento. Algunas de estas aportaciones se enumeran al final de este apartado.

Puesto que las actividades de mantenimiento representan la mayor carga de trabajo durante el ciclo de vida del software, parece evidente que la disponibilidad de herramientas para automatizarlas ayudará a reducir notablemente el coste global del software. Aunque originalmente la mayoría de las herramientas CASE se idearon con otros fines, muchas de ellas pueden utilizarse también para automatizar el PMS. Sin embargo, la mayoría de ellas son herramientas verticales, en el sentido de que automatizan únicamente determinadas tareas del mantenimiento (estimación de costes, pruebas, reestructuración), y existen pocas herramientas horizontales, susceptibles de ser utilizadas a lo largo de todo el proceso. Esta opinión es corroborada por Pigoski (1997), para quien “la ausencia de un proceso de mantenimiento definido dificulta la utilización de herramientas CASE para la gestión del proceso, y las herramientas disponibles sólo son aplicables a actividades concretas del mantenimiento”. La mayoría de las herramientas automáticas para mantenimiento son de reestructuración y reingeniería (por ejemplo, en las ediciones de 1998 y 1999 de la conferencia CSMR se presentaron 15 herramientas, 13 de las cuales eran de estas categorías o eran mejoradas con funcionalidades de estas actividades). Sin embargo, la gran heterogeneidad de los elementos de los sistemas software actuales (programas, bases de datos, documentos, etc.) y la amplia diversidad de cada uno de ellos (multitud de lenguajes de programación; de modelos de datos, etc.), hace que la aplicación de las herramientas que automatizan estas técnicas esté muy restringida por las características del software que se va a modificar.

Entre las herramientas tradicionales para mantenimiento del software se pueden señalar las siguientes: generadores de referencias cruzadas, generadores de organigramas, controladores de código fuente, analizadores automáticos de interfaces, gestores de ficheros, descompiladores, evaluadores del impacto de las modificaciones, y detectores de componentes afectados por los cambios. Estos tipos de herramientas, junto con otros nuevos propuestos en los últimos años, se pueden agrupar en tres categorías (Mazza et al, 1996):

- *De navegación*, que ayudan a encontrar rápida y fácilmente las partes del software que interesan. Ejemplo de este tipo son los “*browsers*” de jerarquías de clases en entornos orientados a objetos.
- *De perfeccionamiento del código fuente* (reformateadores y reestructuradores).
- *De ingeniería inversa*, que procesan el código fuente para producir otro tipo de elemento software de un nivel de abstracción más alto. En esta categoría se incluyen los recuperadores de diseño, redocumentadores, analizadores de código o descompiladores.

En conclusión, a pesar de la gran cantidad de herramientas CASE existentes, las útiles en el PMS son en su mayoría implementaciones de soluciones técnicas que resultan de gran ayuda en tareas concretas, pero no son utilizables como herramientas de gestión del proceso. No obstante, existe la posibilidad de utilizar a lo largo de todo el proceso de mantenimiento algunas herramientas diseñadas con otros fines. Este es el caso de las herramientas para gestión de configuración software. Para profundizar en el tema de herramientas CASE para el PMS se puede consultar el capítulo 6 de Piattini et al (2000).

Otras propuestas formuladas en los últimos años, que plantean ideas interesantes, son las siguientes:

- La arquitectura de una factoría de mantenimiento de software automatizado, basada en la utilización de un meta-entorno de programación, propuesta por Sellink y Verhoef (1999). Estos autores utilizan el meta-entorno ASF+SDF, que permite representar la sintaxis y la semántica de cualquier lenguaje de programación mediante formalismos algebraicos. El problema de esta propuesta es que sólo se ha probado con un único lenguaje (COBOL incluyendo órdenes de SQL embebido). Además, realmente se centra en aspectos puramente técnicos de implementación de los cambios directamente en el código fuente, y no aborda ninguna de las actividades que definen la gestión del PMS.
- La utilización de modelos de actitud ante la tecnología y de capacidad de uso para analizar porqué los ingenieros de mantenimiento usan menos las herramientas de lo que les gustaría (Dishaw y Strong, 1998). La principal conclusión de este trabajo es que los responsables de mantenimiento no están sacando todo el provecho posible de las herramientas. De hecho, se observa que dicho uso es más alto en los casos en que los mantenedores tienen capacidad de decisión propia respecto al uso de herramientas.
- El proyecto industrial Esprit IMPUCT (*Improvement of Maintenance Process Using CASE Tools*), cuyo objetivo es reducir los costes del mantenimiento e incrementar la calidad del software mediante el uso intensivo de herramientas CASE (Chiriatti, 1999). Aunque este proyecto es meramente de innovación, hemos querido citarlo por la gran importancia que concede al uso de herramientas en el PMS.

4.3.8. Medida.

Esta dimensión de la gestión del PMS se refiere a los aspectos del proceso de medida, tal cual lo define el estándar ISO 15504 (apartado 3.3.2). Aunque existen muchos trabajos dedicados a métricas, únicamente incluimos en este estudio los de métricas relacionadas con la gestión del PMS. Las métricas para realizar la estimación del esfuerzo de mantenimiento durante la planificación del proyecto tampoco se incluyen porque ya fueron contempladas en la dimensión de gestión del proyecto (apartado 4.3.2.2). Para profundizar en la aplicación de métricas de producto y de proceso (inicialmente diseñadas para el desarrollo) al mantenimiento se puede consultar el capítulo 5 del libro de Piattini et al (2000).

4.3.8.1. Métricas de Estabilidad del Proceso.

Schneidewind (1999b) ha estudiado una importante faceta de la capacidad del PMS, su estabilidad. Para ello, se ha basado en un conjunto de métricas de fiabilidad y de pruebas, que permiten evaluar la tendencia a lo largo del tiempo del proceso y, en consecuencia, analizar su estabilidad. La estabilidad del proceso se entiende como el incremento en la funcionalidad a la vez que se reducen los fallos, con el paso del tiempo). Por tanto, una pérdida de estabilidad en el proceso afecta negativamente a la fiabilidad del producto. Más concretamente, la investigación ha consistido en buscar las relaciones entre los tres elementos siguientes:

- a) las acciones de mantenimiento, representadas por el número de cambios en el código fuente por KLCF (kilo-líneas de código fuente);
- b) la fiabilidad, representada por varias métricas (número total de fallos, número de fallos que permanecen, y tiempo medio entre fallos); y

- c) el esfuerzo dedicado a las pruebas, representado por el tiempo total dedicado a las pruebas.

El autor también clasifica las métricas en función de si son útiles para medir efectos a corto o a largo plazo. Schneidewind también indica en las conclusiones que su objetivo ha sido proponer un modelo de medida del producto y del proceso unificado, útil tanto para la evaluación del producto como para el análisis de la estabilidad del proceso.

Sin embargo, los resultados del caso real estudiado no permiten sacar conclusiones claras sobre las relaciones entre los tres aspectos antes citados. Además, para este lector la impresión resultante es que la propuesta consiste, en realidad, en un conjunto de métricas que pueden ser útiles para estimar la fiabilidad del PMS, aunque no en todos los casos.

4.3.8.2. Métricas de Mantenibilidad.

Tradicionalmente, la mantenibilidad (o facilidad de mantenimiento) del software se ha cuantificado en términos de atributos del código fuente o de la documentación asociada con él. Estas métricas no se corresponden con la intuición, y de hecho, no producen buenos resultados. Prueba de ello es que no son utilizadas por industria en la estimación de costes del ciclo de vida. En realidad, la mantenibilidad está relacionada con las habilidades del equipo de mantenimiento, las herramientas disponibles, la madurez del proceso, etc.

A partir de esta justificación, Ramage y Bennett (1998) proponen un nuevo modelo para medir la mantenibilidad, que está basado en ver el proceso y la organización de mantenimiento desde un punto de vista sistémico. De esta manera, las cambiantes interrelaciones entre los componentes son un aspecto básico para estimar la mantenibilidad.

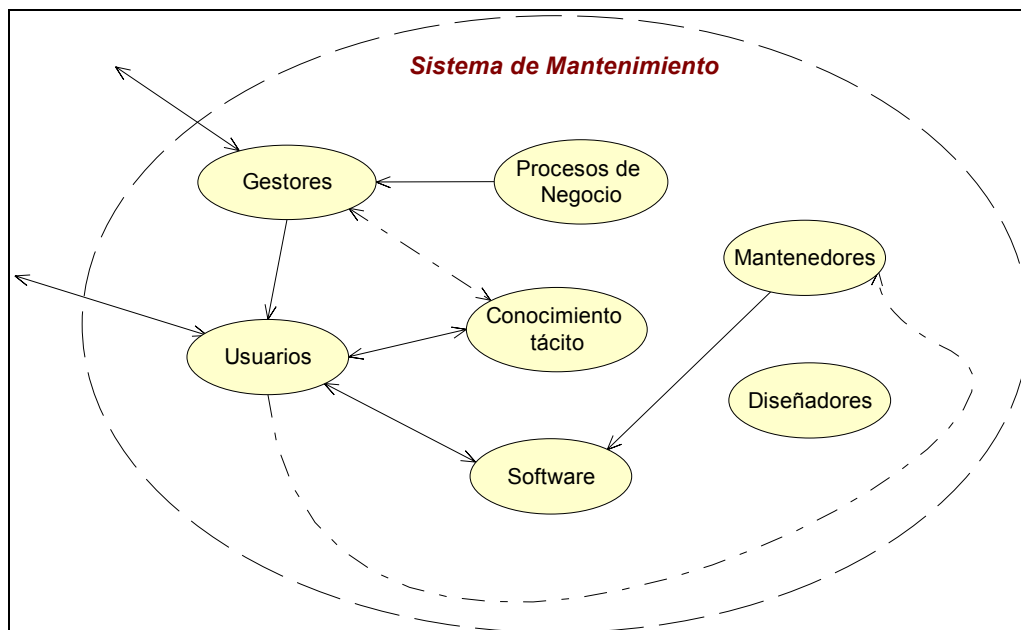


Figura 4-6. Visión sistémica del Mantenimiento: entidades e interrelaciones en un momento determinado.

Para estos autores, el “sistema de mantenimiento” está formado por los tipos de entidades gestores, usuarios, diseñadores, mantenedores, software, procesos de negocio, y conocimiento tácito (ver Figura 4-6). Entre estas entidades existen muchos tipos de interrelaciones, pero no todos están igualmente activados en cada momento. En el artículo citado se describen en detalle todos estos tipos de entidades y todos los tipos de interrelaciones que pueden darse entre ellos, y con el exterior. En un momento determinado sólo un subconjunto de todas las posibles interrelaciones estarán activas, cada una con un grado de activación diferente. Por ejemplo, en la Figura 4-6 se han representado con flechas continuas los tipos de interrelaciones que están activados completamente y con flechas discontinuas los que están parcialmente activos. Los tipos de interrelaciones no representados en la figura no significa que no existan, sino que en el momento actual están inactivos.

Ramaje y Bennett proponen una métrica para cuantificar la mantenibilidad de una pieza de software en un momento determinado que está basada en sus interrelaciones sistémicas. Para ello, a cada interrelación ‘r’ le asignan un cuantificador ‘c_r’, que es un indicador de su grado de activación en cada momento (1 significa completamente desactivada y 10 totalmente activada). Además, cada interrelación también tiene un peso ‘w_r’, que es un indicador de su importancia. El índice de mantenibilidad, IM, para un sistema que tiene n interrelaciones posibles como máximo, viene dado por la fórmula siguiente:

$$IM = \frac{\sum_{r=1}^n c_r \times w_r}{\sum_{r=1}^n 10 \times w_r}$$

Esta propuesta es bastante original pero es incompleta ya que, al menos, falta un método para asignar valor a los cuantificadores c_r y a los pesos w_r. Además, no serviría para estimar la mantenibilidad en términos absolutos sino relativos (su evolución a mejor o a peor). Por ejemplo, sabiendo que el índice de mantenibilidad del elemento software X es 0’65 en el instante t1 no sabemos realmente si eso es bueno o es malo, y en qué medida lo es, pero, si en el instante t2 (habiendo cambiado el estado del sistema) pasa a valer 0’73, entonces podremos decir que ha mejorado.

En cualquier caso, el principal problema es que los autores dan por sentado que la mantenibilidad es directamente proporcional a IM, pero este índice realmente lo que mide es la conectividad de cada elemento del sistema en cada momento, y no está claro que la mantenibilidad sea equivalente a la conectividad.

4.3.8.3. Otras Propuestas.

Stark (1996) plantea que realizar mediciones es fundamental para una correcta gestión del PMS. En concreto, este autor presenta su experiencia en diseñar e implantar un proceso de medida basado en el paradigma GQM (en línea con la posterior propuesta de Niessink comentada en el apartado 4.3.5.1.1). Stark comenta que, en su organización, han necesitado implantar un proceso de medida para que los gestores e ingenieros puedan comprender y gestionar mejor el esfuerzo de mantenimiento. Los positivos resultados de la experiencia han permitido que dichos gestores e ingenieros utilicen las métricas para tres fines diferentes:

- Atención directa, es decir, ¿A qué problemas tengo que prestarles atención?;

- Resolver problemas, o sea, ¿Qué opciones podría yo elegir?; y
- Registro de puntos: ¿Cómo lo estoy haciendo?.

El autor explica que, aplicando GQM, se establecieron tres objetivos: maximizar la satisfacción del cliente, minimizar costes, y minimizar tiempos. A partir de ellos surgieron 10 preguntas y por último, 14 métricas.

Lam et al (1999) proponen un marco de trabajo para el proceso de medida orientado a ayudar en la gestión de los cambios en los requisitos. El marco de trabajo establece cuatro asuntos de interés: planificar los cambios, estimar el impacto de los cambios, determinar la volatilidad de los requisitos, e implementar los cambios de forma eficiente. Para cada uno de ellos, el marco de trabajo incorpora los siguientes apartados:

- Escenario del problema: La situación (no deseada) que podría ocurrir en un proyecto si no se realiza un control del proceso de medida para ese asunto.
- Indicadores: Lista de indicadores que pueden utilizarse por los gestores para aliviar y evitar el escenario del problema.
- Usos: Los usos de gestión de requisitos específicos para los que dichos indicadores pueden servir.
- Planes de acción genéricos: Guías prácticas sobre cómo utilizar y aplicar los indicadores como base para acciones de gestión.

Lehman et al (1998) han calculado los valores de un conjunto de métricas de evolución en el mantenimiento de varios productos software reales. Las métricas utilizadas han sido las siguientes:

- Número de secuencia de versiones¹⁴ o *releases* (sistema de cómputo del tiempo).
- Tamaño del sistema – subsistemas, módulos, archivos, etc.
- Elementos manejados – subsistemas, módulos, archivos, etc.
- Elementos añadidos – subsistemas, módulos, archivos, etc.
- Elementos cambiados – subsistemas, módulos, archivos, etc.
- Elementos eliminados – subsistemas, módulos, archivos, etc.
- Número de actuaciones sobre elementos - subsistemas, módulos, archivos, etc.
- Tiempo entre *releases*.
- Esfuerzo aplicado (en las unidades apropiadas).
- Defectos encontrados antes y después de una versión – por *release*.
- Defectos corregidos antes y después de una versión – por *release*.

Los resultados del estudio han confirmado las postulados de las, ya clásicas, “*Leyes de la evolución del software*” (Lehman, 1980). Más en concreto, los autores dicen que las métricas de estimación combinadas con modelos lineales de análisis han confirmado su utilidad para estimar los efectos de las políticas de mejora del PMS sobre el comportamiento global del software a largo plazo.

¹⁴ Se considera que una versión se diferencia de otra en cambios importantes en su funcionalidad. En cambio, las *releases* sólo tienen pequeños cambios, normalmente debidos a la corrección de errores detectados.

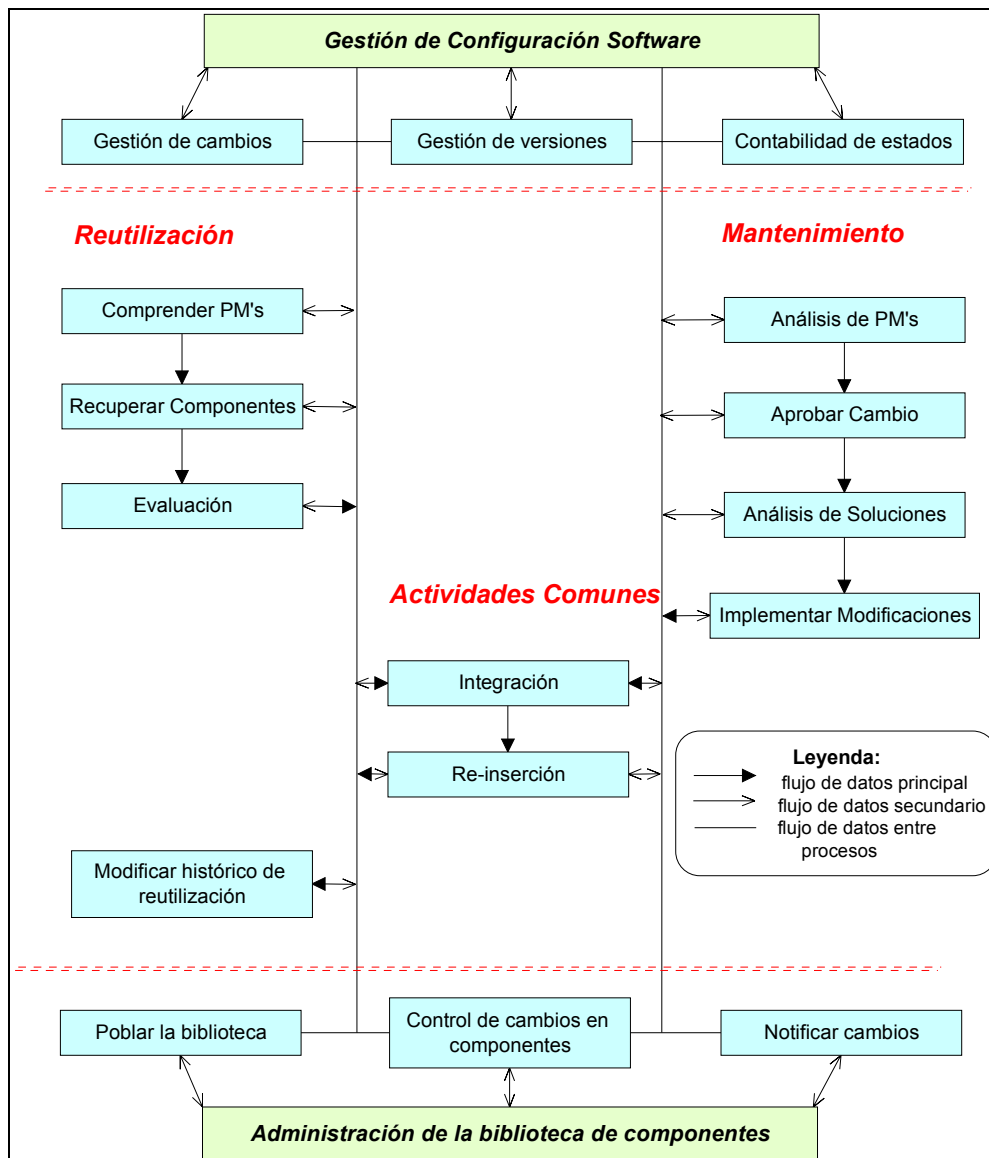
Fioravanti et al (1999) han estudiado la utilidad de un conjunto de métricas clásicas (diseñadas para estimación de esfuerzo en proyectos de desarrollo de software orientado a objetos) en la estimación del esfuerzo de mantenimiento adaptativo. Las principales lecciones aprendidas por los autores son: durante el mantenimiento adaptativo debe prestarse atención a la definición de los métodos (en especialmente a su interfaz); la métrica más útil ha resultado ser una adaptación de la clásica para medir la complejidad/tamaño de las clases; y el factor más importante para estimar el esfuerzo de mantenimiento adaptativo es la complejidad y el número de los interfaces para los métodos definidos localmente.

4.3.9. Reutilización.

En los últimos años, la reutilización ha pasado a considerarse la mejor solución para mejorar la productividad de un equipo de desarrollo de software y para reducir los costes de mantenimiento. A pesar de esta importancia, prácticamente no existen publicaciones relacionando el proceso de reutilización y el PMS; probablemente porque la problemática de la reutilización es muy similar en desarrollo y mantenimiento y las propuestas son casi siempre útiles para ambos tipos de proyectos.

Centrándonos en el ámbito de esta tesis, la dimensión de gestión de la reutilización para el PMS se refiere a la promoción de la reutilización de productos de trabajo software desde una perspectiva organizacional y de producto/proceso. En este sentido, la única publicación conocida es la escrita por Kown et al (1999), con la propuesta conocida como “Mantenimiento con Reutilización” (*Maintenance with Reuse*, MwR). MwR es un modelo de proceso integrado para el mantenimiento de un sistema software heredado (*legacy software*) mediante actividades de reutilización y de Gestión de la Configuración Software” (GCS). El modelo está orientado a la reutilización de una biblioteca de componentes software e integra cuatro procesos: GCS, reutilización, mantenimiento y administración (de la biblioteca de componentes reutilizables). La Figura 4-7 resume este modelo de procesos.

La opinión del autor de esta tesis es que este modelo de procesos está inmaduro. En realidad, la única aportación concreta del artículo citado es un prototipo de herramienta, llamada TERRA, para dar soporte al proceso de administración de la biblioteca de componentes. Aparte del problema de que la nomenclatura y la representación gráfica son algo confusas, la propuesta adolece de falta de estandarización. En este sentido, se hecha a faltar su adecuación al ciclo de vida del software propuesto por ISO (por ejemplo), de manera que quede claro cuál es el proceso principal (se supone que el de mantenimiento) y que papel desempeñan los procesos de soporte (GCS) y organizacionales (reutilización y administración de la biblioteca de componentes – este último es un caso particular del proceso de gestión).

Figura 4-7. Modelo de procesos de MwR (*Maintenance with Reuse*).

4.3.10. Alineamiento Organizacional.

La única contribución conocida en esta dimensión es el reciente capítulo de Chapin (2002) incluido en el libro dedicado a la gestión del mantenimiento ya comentado antes. Este autor realiza un estudio sobre la relación del PMS y la fortaleza y desenvoltura organizacionales, que está fundamentado en una teoría organizacional sistemática. Describimos a continuación un resumen de esta teoría.

En las *organizaciones* (humanas), los *sistemas* determinan la manera en que ciertas situaciones recurrentes pueden ser manejadas mediante *métodos*, los cuales combinan servicios de personal, máquinas y materiales operando de forma conjunta. Los *sistemas de información* son de especial interés en esta teoría ya que suelen incluir software que implementa los métodos. En un sistema, una parte importante de los métodos son las opciones de gestión de las *reglas de negocio*, las cuales tienen dos componentes importantes: decisión y acción. Normalmente, las reglas de negocio están anidadas y se implementan dentro de los sistemas. En

consecuencia, los sistemas suelen tener componentes más simples, llamados subsistemas. Los sistemas y sus reglas de negocio son los contribuyentes básicos a la capacidad de rendimiento operacional de una organización, es decir, a la salud y desenvoltura organizacionales.

Para cambiar la salud y desenvoltura de una organización, la gestión tiene que especificar los cambios que se deben llevar a cabo en cada uno de los sistemas organizacionales. Habitualmente, cambiar un sistema existente requiere cambiar las reglas de negocio del sistema, lo que suele implicar, a su vez, cambios en los servicios proporcionados por el personal, máquinas, materiales y métodos. Los cambios de personal suelen ser los más sensibles y complicados de manejar y, debido a su característica de fractalidad (similar comportamiento a diferentes escalas), pueden surgir al hacer cambios a las reglas de negocio, al sistema, o a la organización

La gestión desempeña un papel clave en la selección, dirección y especificación de cambios en una organización, sistema o regla de negocio. Llevar a cabo los cambios requeridos por la gestión suele exigir que el personal de sistemas haga un esfuerzo adicional. En el caso de sistemas de información, el personal de estos sistemas debe estar prevenido porque dichos cambios implican cambios adicionales, que a veces pueden ser bastante extensos. Los sistemas de información automatizados (informáticos) suelen ser cambiados mediante procesos de mantenimiento de software; y eso a pesar de que dichos sistemas incluyen, además del software, otro tipo de elementos como personal, máquinas (diferentes de los computadores), materiales y métodos. De lo anterior, se deduce que la gestión del PMS implica más cosas que la mera gestión de los cambios en el software. En conclusión, el PMS cambia los sistemas y, por tanto, también cambia el rendimiento operacional de las organizaciones.

Como complemento a la anterior teoría, Chapin también incluye veinte lecciones aprendidas, de las cuales 10 son actividades de mantenimiento que tienen efectos potenciales negativos sobre la salud y desenvoltura organizacional, mientras que las otras 10 tienen efectos potenciales positivos.

5. El Entorno MANTIS

En este capítulo se incluye la parte nuclear de este trabajo de tesis. En el capítulo de introducción ya se comentó que el objetivo principal de esta tesis, la definición de un entorno extendido de ingeniería del software para la gestión integral de proyectos de mantenimiento, implica a muchos aspectos interrelacionados que tienen diferente naturaleza (conceptual, metodológica, instrumental, etc.). Por tanto, nos hemos enfrentado a un problema polifacético y, por ello, ha sido necesario analizar la problemática de gestionar proyectos de mantenimiento desde una perspectiva holística, es decir, intentando integrar los diversos aspectos desde los puntos de vista conceptual, metodológico y tecnológico.

El capítulo tiene dos partes diferenciadas. En la primera parte (apartados 5.1 y 5.2) se realiza una introducción a los objetivos y características del Entorno MANTIS, y se presentan los tres diferentes tipos de elementos o herramientas que forman este “entorno extendido”. En la segunda parte (formada por los restantes apartados), se presenta el marco de trabajo conceptual definido, que es la pieza fundamental del Entorno y necesaria, además, para poder integrar los métodos y técnicas (capítulo 6) y las herramientas software (capítulo 7).

5.1. Introducción.

Es bien conocido que el mantenimiento es la etapa más costosa del ciclo de vida del software. Además, la tendencia en los últimos años ha sido creciente de forma que, en bastantes organizaciones, el mantenimiento se lleva más de dos tercios de los recursos consumidos a lo largo de la vida útil de un producto software (Piattini et al, 2000). Por esta razón, es importante abordar este proceso desde nuevas perspectivas que permitan mejorar la situación.

Además, en el capítulo 1 ya se ha justificado la necesidad de disponer de herramientas adaptadas a las características especiales del PMS, como son el entorno MANTIS y las herramientas CASE incluidas en él.

En esta línea, esta tesis pretende aportar al PMS dos aproximaciones nuevas, una de carácter gerencial y otra de carácter tecnológico, que usadas de forma conjunta ayuden a gestionar mejor los proyectos de mantenimiento de software. Estas dos aproximaciones generales, que se comentan a continuación, son:

- Abordar el Proceso de Mantenimiento del Software como un proceso de negocio.
- Utilizar un Entorno de Ingeniería del Software diseñado “ex profeso” para gestionar este proceso.

Además, el Entorno MANTIS pretende ser una ayuda para gestionar la complejidad inherente al PMS, concediendo especial atención a los dos aspectos siguientes:

- Integración del conocimiento, para lo cual se ha definido un marco de trabajo conceptual y metodológico adecuado; y
- Orientación a la automatización por medio de herramientas software, para lo cual se han definido una arquitectura tecnológica (basada fundamentalmente en diversas propuestas de estándares oficiales o “de facto” internacionales) y una colección de herramientas CASE para soportar algunas de las actividades de gestión del PMS.

5.1.1. El Mantenimiento como Proceso de Negocio.

Un proceso de negocio es “una colección de tareas de trabajo interrelacionadas, iniciadas en respuesta a un evento, que permiten alcanzar un resultado específico para el cliente del proceso” (Sharp y McDermott, 2001). Es por tanto un concepto bastante general en el cual es posible englobar procesos más específicos. En esta línea, algunos trabajos recientes proponen abordar los procesos software desde una perspectiva de procesos de negocio más amplia que la de un mero proceso tecnológico. Cockburn (2000) llama al resultado de aplicar esta idea una “*Big-M Methodology*”. En el Entorno MANTIS se ha aplicado esta misma idea para definir un entorno integrado que permita abordar los proyectos de mantenimiento de software con una perspectiva de proceso de negocio (Ruiz y Piattini, 2001) (Ruiz et al, 2002), incluyendo, entre otros, los aspectos siguientes (ver Figura 5-1):

- Las personas con ciertas habilidades desempeñan ciertos roles en el proyecto trabajando juntas en diversos equipos (grupos de personas).
- Estas personas utilizan técnicas (metodologías) para construir productos que siguen ciertos estándares (normas) y satisfacen medidas (criterios) de calidad. Los procesos también deben satisfacer criterios de calidad.

- Las técnicas requieren ciertas habilidades y herramientas; las herramientas ayudan a cumplir los estándares.
- Los equipos participan en actividades que pertenecen a procesos englobados dentro del proyecto; cada actividad realizada ayuda a alcanzar hitos significativos que indican cómo avanzan los procesos.

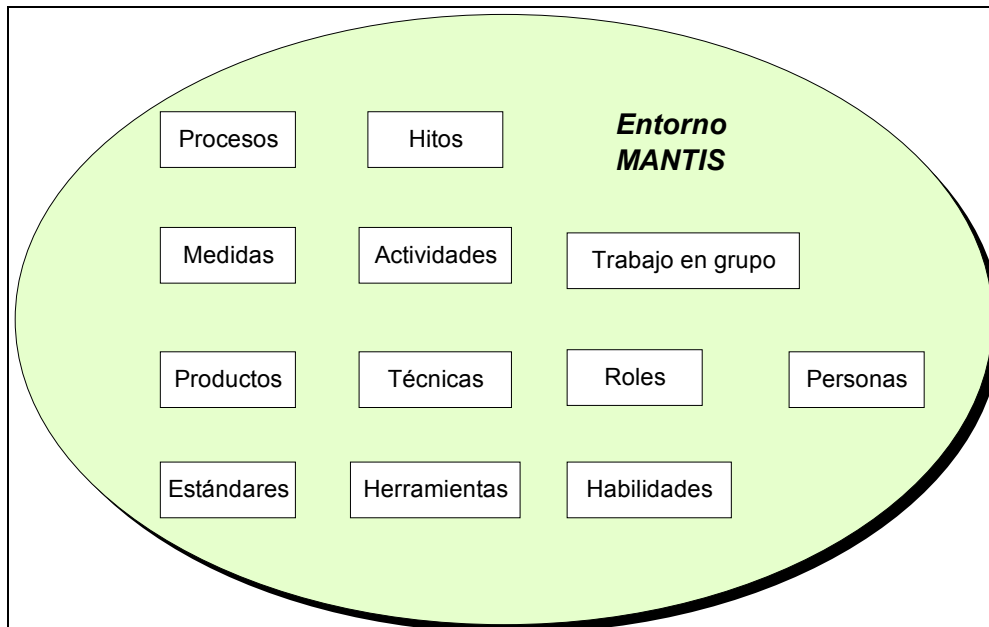


Figura 5-1. Aspectos del mantenimiento como proceso de negocio.

Esta idea de conceder al mantenimiento del software (y de forma más general, al mantenimiento de sistemas de información) la categoría de proceso de negocio está subyacente en algunos trabajos recientes. Entre ellas merece destacarse la reciente propuesta de Rajlich y Bennet (2000) que proponen “un nuevo ciclo de vida del software en el cual el mantenimiento está formado por varias etapas distintas, cada una con diferentes actividades, herramientas y consecuencias de negocio”. Estos autores se centran en el punto de vista de los usuarios en vez de en el punto de vista del desarrollador (que es lo que hacen otros modelos de ciclo de vida tradicionales: espiral, cascada, etc.). Por ello, Rajlich y Bennet consideran que, desde una perspectiva de negocio, un producto software pasa por cinco etapas diferentes:

- *Desarrollo inicial*: los ingenieros construyen la primera versión operativa del producto software.
- *Evolución*: los ingenieros extienden las capacidades del producto software para satisfacer las necesidades cambiantes de los usuarios. En esta etapa son habituales los cambios iterativos, modificaciones y eliminaciones en la funcionalidad del software.
- *Servicio* (o saturación): los ingenieros reparan pequeños defectos y llevan a cabo cambios funcionales simples. Durante esta etapa los cambios son, a la vez, caros y difíciles porque la arquitectura software adecuada y las habilidades del equipo de trabajo se van perdiendo.
- *Retirada* (o declinar): la organización decide no soportar más servicios, aunque buscan generar ingresos u otros beneficios del producto software, que ya no cambia, tanto tiempo como sea posible.

- *Cierre*: la organización da de baja el producto de su catálogo y orienta a los usuarios hacia nuevos productos, si los hubiese.

Varias características del software y del equipo de desarrollo cambian substancialmente de una fase a otra: nivel de experiencia del equipo de trabajo, arquitectura del software, decaimiento del software (es decir, la realimentación positiva entre la pérdida de nivel de experiencia del personal y la pérdida de coherencia arquitectónica) y costes económicos. Desde el punto de vista del proceso de mantenimiento del software (PMS), otra diferencia importante entre una fase y otra es la frecuencia diferente con que cada tipo de mantenimiento (ver capítulo 3) se lleva a cabo. El mantenimiento correctivo (corregir los errores) es más usual en la etapa de servicio mientras el mantenimiento perfectivo (hacer cambios a la funcionalidad) es más frecuente en la etapa de evolución. Los otros dos tipos de mantenimiento, tal como son definidos por las normas ISO/IEC (1998e), es decir, adaptativo (provocado por cambios en el entorno) y preventivo (hacer cambios para mejorar las propiedades de calidad y evitar problemas futuros) suelen ser bastante menos frecuentes.

En resumen, cada una de dichas etapas difiere de las demás en algunos aspectos esenciales. Además, mientras que la etapa de desarrollo inicial está bien documentada y existen numerosos métodos, técnicas y herramientas, las otras cuatro etapas han sido estudiadas y analizadas en menor grado. Por tanto, puesto que el PMS ocurre durante las cuatro últimas etapas citadas, es necesario desarrollar métodos, herramientas y entornos específicos, distintos de los disponibles para el proceso de desarrollo (Piattini et al, 2000). La propuesta que presentamos en esta tesis está orientada a las cuatro etapas posteriores al desarrollo inicial, y especialmente a las etapas de evolución y servicio que son las más costosas en mantenimiento.

5.1.2. Un Entorno Extendido.

Decimos que MANTIS es un Entorno extendido (y utilizamos la palabra entorno con E mayúscula para denotarlo) porque extiende e integra los dos conceptos fundamentales siguientes:

- *Metodología*, entendida en el sentido amplio propuesto por Graham et al (1997), es decir, formada no solo por un conjunto de métodos o técnicas relacionados, sino también por un modelo de proceso(s), entregables, métricas, guías de gestión (incluyendo roles y organización del equipo de trabajo), y herramientas.
- *Entorno de Ingeniería del Software*, entendido tal como se definió en el capítulo 3, es decir, como una colección de herramientas software utilizadas para soportar actividades de ingeniería del software (en este caso las actividades que forman un proyecto de mantenimiento de software).

Adicionalmente, el Entorno MANTIS también se puede considerar un EIS de dominio específico, tal como lo definen Oliveira et al (1998), ya que incluye una serie de elementos en varios niveles de abstracción incluyendo el conocimiento del dominio. En MANTIS este conocimiento del dominio se refiere a la gestión de proyectos de mantenimiento de software, que está incorporado mediante un conjunto de ontologías integradas (ver apartado 5.5).

5.1.3. Características Generales de MANTIS.

Las principales características que definen y resumen la naturaleza del Entorno MANTIS son las siguientes (Ruiz y Piattini, 2001), (Ruiz et al, 2002a), (Ruiz et al, 2002b):

a) Integración por Medio de Herramientas:

Los servicios de MANTIS son provistos mediante herramientas software siguiendo la filosofía de los EIS integrados. La utilización integrada de las herramientas es de gran ayuda para mejorar la productividad, reducir las posibilidades de error y facilitar la gestión, supervisión y control. Aunque existen diversas propuestas de EIS con estos objetivos (PSEE, PCTE, ECMA) que han sido presentadas en el capítulo 3, todas ellas están pensadas de forma general para todos los procesos software. Por ello, aunque incluyen contribuciones y modelos que pueden utilizarse de forma genérica en el PMS, ninguna de ellas tiene en cuenta las características especiales del mantenimiento.

En MANTIS se aborda dicha integración desde las cuatro dimensiones comentadas en el capítulo 3:

- *Datos*, utilizando las técnicas tradicionales basadas en la definición de un repositorio compartido y usando estándares para el formato de los datos y metadatos.
- *Control*, utilizando el soporte externo de un Sistema de Gestión de flujos de Trabajo.
- *Presentación*, definiendo un interfaz común para que los usuarios puedan interaccionar con las diversas herramientas integradas en el Entorno o invocar las herramientas externas.
- *Conocimiento*, definiendo una colección de ontologías para la gestión de proyectos de mantenimiento de software e implementando dicho conocimiento mediante diversos metamodelos y modelos que instancian los anteriores, que se almacenan en el repositorio compartido.

b) Orientación a Procesos:

La orientación a procesos es un vehículo de integración muy importante en los proyectos y actividades de ingeniería del software (Randall y Ett, 1995). En MANTIS se contemplan dos formas de integración:

- de los procesos de la organización con el Entorno, mediante la definición, dentro de éste, de un marco de trabajo conceptual que contempla la idea de proceso como una de sus partes fundamentales;
- y de las herramientas con los procesos, que está basada en la propuesta PSEE (*Process-centered Software Engineering Environment*) que se ha comentado en el capítulo 3. A tal efecto, en el capítulo 7 se detallan la arquitectura software y los tipos de herramientas software.

Debe destacarse que esta orientación a procesos no implica que el entorno tenga que proveer herramientas tanto para el modelado como para el soporte (implementación de la reificación) de los procesos. Tal como indica Kobialka (1999), ambos aspectos son diferentes y se deben resolver de forma separada. De esta manera, en MANTIS se han incluido un marco conceptual y varias herramientas para el modelado y representación estandarizada de los procesos. En cambio, el control de la reificación de los procesos, es decir, el seguimiento de la

ejecución de los proyectos, se ha excluido del alcance del trabajo al considerar que esta función la pueden desempeñar perfectamente los Sistemas de Gestión de Flujos de Trabajo (SGFT's).

c) Especialización en Mantenimiento:

Esta especialización se concreta en la utilización de métodos y técnicas específicos para el PMS. Este es el caso de la metodología MANTEMA y de otras técnicas y métodos, especialmente desarrollados o adaptados, que se comentan en el capítulo 6. La metodología MANTEMA fue el principal resultado del proyecto del mismo nombre, el primero de los tres proyectos de I+D que han justificado de forma pragmática el desarrollo de esta tesis (ver capítulo 1). En el anexo B se puede consultar un resumen de dicha metodología. Para mayor profundización también se puede consultar la tesis doctoral de Macario Polo (2000).

d) Escalabilidad y Adaptabilidad:

La escalabilidad, es decir, su utilidad en proyectos de mantenimiento de software de cualquier tamaño se basa, fundamentalmente, en el uso de un modelo de roles y responsabilidades que permite una gran variedad de estructuras funcionales y organizacionales para llevar a cabo los proyectos. Ello es posible gracias a contemplar un modelo organizacional abierto y flexible, tanto en el marco de trabajo conceptual (ver apartado 5.5.3.4 con la sub-ontología de los agentes) como en la metodología MANTEMA y demás técnicas (ver anexo B).

La adaptabilidad se contempla mediante la utilización del proceso de adaptación (*tayloring*) definido en la norma ISO 12207 (ISO/IEC, 1995). Usando este proceso como guía, es posible “adaptar” el modelo de procesos definido en MANTIS a las necesidades y características de cada organización y/o proyecto.

5.2. Estructura General.

En la Figura 5-2 se muestra un esquema de los componentes principales del Entorno MANTIS. Como se comentó anteriormente, MANTIS no es un EIS en el sentido habitual, ya que, además de aplicaciones software, ha sido extendido para incluir aspectos conceptuales y metodológicos. De todas maneras, se ha considerado adecuado respetar la nomenclatura de herramientas para todos sus componentes por razones de homogeneidad y porque, en sentido amplio, realmente lo son.

5.2.1. Clases de Herramientas.

Como muestra la Figura 5-2, el Entorno MANTIS integra tres tipos de elementos o herramientas: conceptuales, metodológicas y técnicas (software). A continuación se presentan brevemente cada una de estas tres clases, que son desarrolladas, respectivamente, en los capítulos 5, 6 y 7.

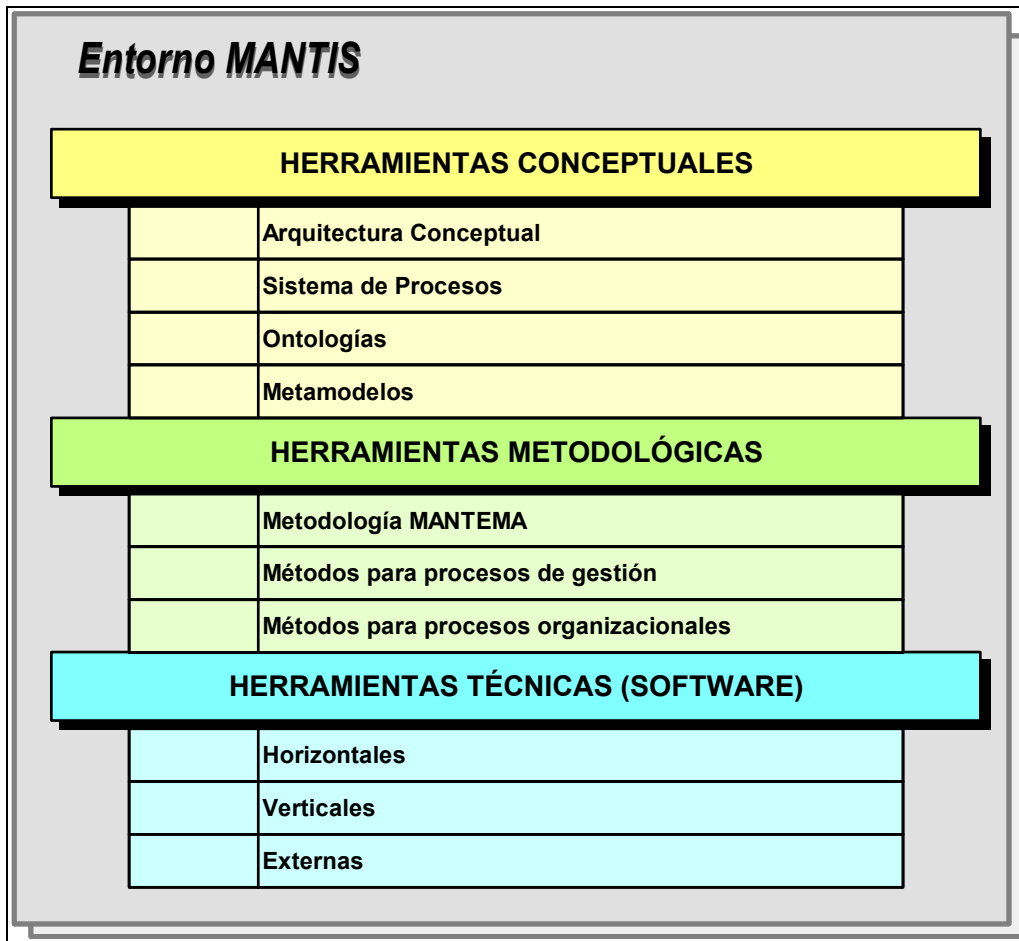


Figura 5-2. Estructura General del Entorno MANTIS.

5.2.1.1. Herramientas Conceptuales.

Las herramientas conceptuales establecen un marco de trabajo adecuado para gestionar el PMS teniendo en cuenta los dos intereses comentados en el apartado 5.1, integración del conocimiento y orientación a la automatización. Para ello, MANTIS incluye las siguientes herramientas conceptuales:

- Una arquitectura conceptual multinivel para poder manejar la complejidad y diversidad de todos los datos y metadatos que se producen y utilizan en proyectos de mantenimiento (Ruiz et al, 2001c).
- Una ontología (organizada en varias subontologías) que permite compartir el conocimiento del dominio del problema (el mantenimiento de software) y usar una nomenclatura común (Ruiz et al, 2003).
- Un sistema de procesos que identifica los procesos (y sus características) que se tienen que realizar cuando se llevan a cabo proyectos de mantenimiento de software. Este sistema está basado en los estándares de ISO comentados en el capítulo 3.
- Un metamodelo del proceso de mantenimiento (formado por varios metamodelos parciales) que permite manejar diferentes modelos del PMS y sus procesos auxiliares organizativos y de gestión (Ruiz et al, 2001a), (Ruiz et al, 2001b), (Márquez et al, 2001a).

Los apartados 5.3-5.6 de este mismo capítulo están dedicados a explicar cada una de estas herramientas conceptuales, incluyendo su utilidad y características.

5.2.1.2. Herramientas Metodológicas.

Las herramientas metodológicas son el conjunto de métodos y técnicas desarrollados “*ex profeso*” para el Entorno MANTIS o integrados en él por su utilidad a la gestión del mantenimiento aunque se definieran pensando en otra utilidad. Las herramientas metodológicas son de 3 clases:

- La metodología MANTEMA, diseñada completa y especialmente para tener en cuenta las especiales características que diferencian el proceso de mantenimiento del software del de desarrollo (Piattini et al, 2000).
- Una colección de métodos para los procesos de gestión es decir, la gestión global del proceso, la gestión del proyecto, la gestión de la calidad y la gestión de los riesgos. Estos procesos son los definidos como tales en el estándar ISO 15504 (1998c) (ver capítulo 3). Ejemplos de estos métodos y técnicas (todos ellos referidos a proyectos de mantenimiento de software) son los siguientes: externalización de servicios, gestión del conocimiento, estimación de recursos, distribución de recursos en una cartera de proyectos, auditoría y control de un proyecto, factores de riesgo, y objetivos de control.
- Dada la importancia que el Entorno MANTIS concede a los aspectos organizacionales (cuya fundamentación es considerar el mantenimiento del software como un proceso de negocio), también se ha definido otra colección de métodos para los procesos organizacionales, es decir, la mejora del proceso, la gestión de los recursos humanos, la gestión de las infraestructuras (recursos no humanos) y la medida. Estos procesos son los incluidos en la subcategoría organizacional en la norma ISO 15504. Un ejemplo de estos métodos y técnicas son las diversas métricas definidas para gestionar el mantenimiento.

Estos métodos y técnicas son presentados en el capítulo 6, salvo la metodología MANTEMA que, por las razones ya comentadas anteriormente, sólo se incluye como resumen en el anexo B.

5.2.1.3. Herramientas Técnicas (Software).

Una de las características fundamentales del Entorno MANTIS es su orientación a la automatización mediante el uso de herramientas software. Por esta razón, se han incorporado actividades automatizadas en el marco conceptual (en la ontología de los flujos de trabajo y en los metamodelos) y se facilita el uso integrado de todas las herramientas software al seguir los siguientes principios arquitecturales de la propuesta PSEE (Derniame et al, 1999):

- Utilizar un motor de procesos (*process engine*) para controlar el flujo de información entre los actores o realizadores, de acuerdo con los modelos de procesos definidos. El Entorno MANTIS utiliza un SGFT (Sistema de Gestión de Flujos de Trabajo) para tal fin, ya que se ha considerado que este tipo de herramientas cumplen satisfactoriamente esta necesidad. Además, en la actualidad ya existen suficientes productos comerciales de este

tipo lo que supone una ventaja pragmática frente a prototipos más específicos de motores de proceso para EIS.

- Almacenar en un repositorio global el metamodelo del proceso junto con la definición del producto (el software) y la información de reificación del proceso (los datos de planificación y seguimiento de los proyectos).
- Permitir la compartición de datos y metadatos con otros sistemas y herramientas por medio de servicios de importación y exportación que utilizan un adecuado formato de comunicación.

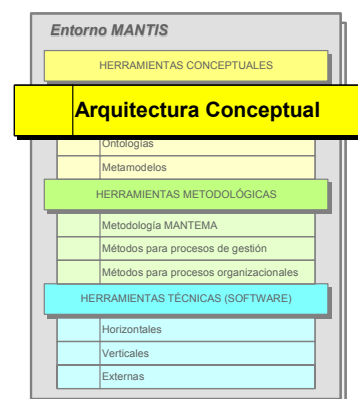
La arquitectura software de MANTIS y la descripción de sus principales herramientas técnicas se presentan en el capítulo 7. Estas herramientas están agrupadas, según su papel e integración en el Entorno, en tres tipos diferentes:

- Verticales, que dan servicio para automatizar actividades de alguno de los procesos de gestión u organizacionales, o al propio PMS. Algunas herramientas de este grupo son MANTOOL (gestor de peticiones de mantenimiento utilizando la metodología MANTEMA), METAMOD (modelado y metamodelado de los procesos), MANTICA (cálculo de métricas de calidad), SREM (estimación de recursos), CREM (gestión de recursos).
- Horizontales, que dan servicios de carácter general al Entorno, es decir, a otras herramientas verticales u horizontales. Ejemplos de estas herramientas son MANTIS-Tool (interfaz de usuario integrado), RepManager (gestor del repositorio XML/XMI de datos y metadatos) y KM-MANTIS (gestor de la base de conocimiento).
- Por último se incluyen las herramientas externas, que son aquellas que, sin formar parte del Entorno MANTIS, pueden ser utilizadas desde dentro de el (es decir, por sus herramientas verticales y horizontales), aprovechando las capacidades de intercambio y comunicación de datos y metadatos que provee el Entorno. La principal herramienta de este tipo son los Sistemas de Gestión de Flujos de Trabajo (SGFT's), ya que MANTIS delega en uno de ellos la responsabilidad en la reificación de los procesos, es decir, el seguimiento y captura de los datos referidos a lo que ocurre durante la realización de los proyectos de mantenimiento.

5.3. Arquitectura Conceptual.

Un principio importante de la ingeniería del software moderna, para gestionar la complejidad, es la separación de un sistema en capas de encapsulación, que pueden especificarse, diseñarse y construirse de manera independiente (al menos en gran parte). Además, la gestión de meta-información es fundamental en entornos abiertos y heterogéneos (Crawley et al, 1997a). Con esta filosofía, en MANTIS se definen 4 niveles conceptuales (ver Tabla 5-1) que están basados en el estándar MOF (*Meta-Object Facility*) para metamodelización con orientación a objetos, propuesto por el *Object Management Group*, OMG (2002b).

En el anexo C se ha incluido una explicación detallada



de la arquitectura de niveles de MOF y, en especial, de las características del Modelo MOF, que es el meta-metamodelo orientado a objetos incluido en el nivel superior M3 para permitir la integración de metamodelos diferentes y poder representar todos ellos mediante una misma colección de constructores común. Un resumen de las ventajas de incluir el nivel M3 orientado a objetos y autodefinido se puede consultar en (Crawley et al, 1997b).

Nivel	MANTIS	MOF
M3	Modelo MOF	Modelo MOF (meta-metamodelo)
M2	Metamodelos del PMS	Meta-modelos
M1	Modelos de proceso de MANTEMA y otros métodos y técnicas para procesos de gestión y organizacionales	Modelos
M0	Datos de proyectos de mantenimiento	Datos

Tabla 5-1. Niveles de la arquitectura conceptual basada en MOF.

En el nivel M0 están los datos de proyectos reales y concretos de mantenimiento del software con restricciones de tiempo, costes, etc. Los datos manejados en este nivel son instancias de los conceptos definidos en el nivel superior M1. Los modelos utilizados en el nivel M1 están basado en la metodología MANTEMA y en un conjunto de métodos y técnicas de soporte a los procesos de gestión y organizacionales, que han sido adaptados a las particularidades del mantenimiento. El nivel M2 se corresponde con el metamodelo general del PMS (completado con algún otro metamodelo), que es generalizado mediante el Modelo MOF (ver anexo C) en el nivel M3. De esta manera, todos los conceptos representados en los metamodelos del nivel M2 se consideran ejemplares (instancias) de los constructores Clase-MOF o Asociación-MOF del nivel M3. Por ejemplo, conceptos como "Actividad de Mantenimiento", "Recurso" o "Artefacto" son ejemplares de Clase-MOF y "Actividad usa Recurso" o "Artefacto es_entrada_de Actividad" son ejemplares de Asociación-MOF. El nivel M3 permite llegar a la mayor abstracción posible ya que, aunque se trabaje con diferentes modelos y metamodelos, como es el caso en MANTIS, todos ellos se pueden representar utilizando el Modelo MOF.

En la Figura 5-3 se muestra un ejemplo con correspondencias entre los 4 niveles. En el nivel M0 se encuentran los datos de un determinado proyecto de mantenimiento que está siendo gestionado mediante el Entorno MANTIS. Estos datos incluyen la información de la intervención de mantenimiento correctivo urgente nº 36 cuyos principales datos son: la aplicación a modificar es 'Transfer', la petición de modificación la hizo 'Martín Canovas' en diciembre-2000, la aprobó 'MPV' en febrero-2001, el responsable de su implementación fue 'Luis García', y los recursos consumidos fueron '40 horas de programador'. Esta intervención y todas las demás del mismo tipo son instancias de la clase 'Intervención Correctiva Urgente' que pertenece al modelo 'MANTEMA' del nivel M1, es decir, es una actividad de MANTEMA cuyo identificador es NP2 (ver anexo B para más detalle sobre las actividades definidas en la metodología MANTEMA). A su vez, 'Intervención Correctiva Urgente' y los demás tipos de actividades y tareas definidos en MANTEMA (y en el resto de métodos y técnicas utilizados) son instancias de la clase "Actividad" perteneciente al metamodelo del PMS del nivel M2. Por último, "Actividad" y demás conceptos del metamodelo, definidos como metaobjetos, son instancias de Clase-MOF en el Modelo MOF del nivel M3.

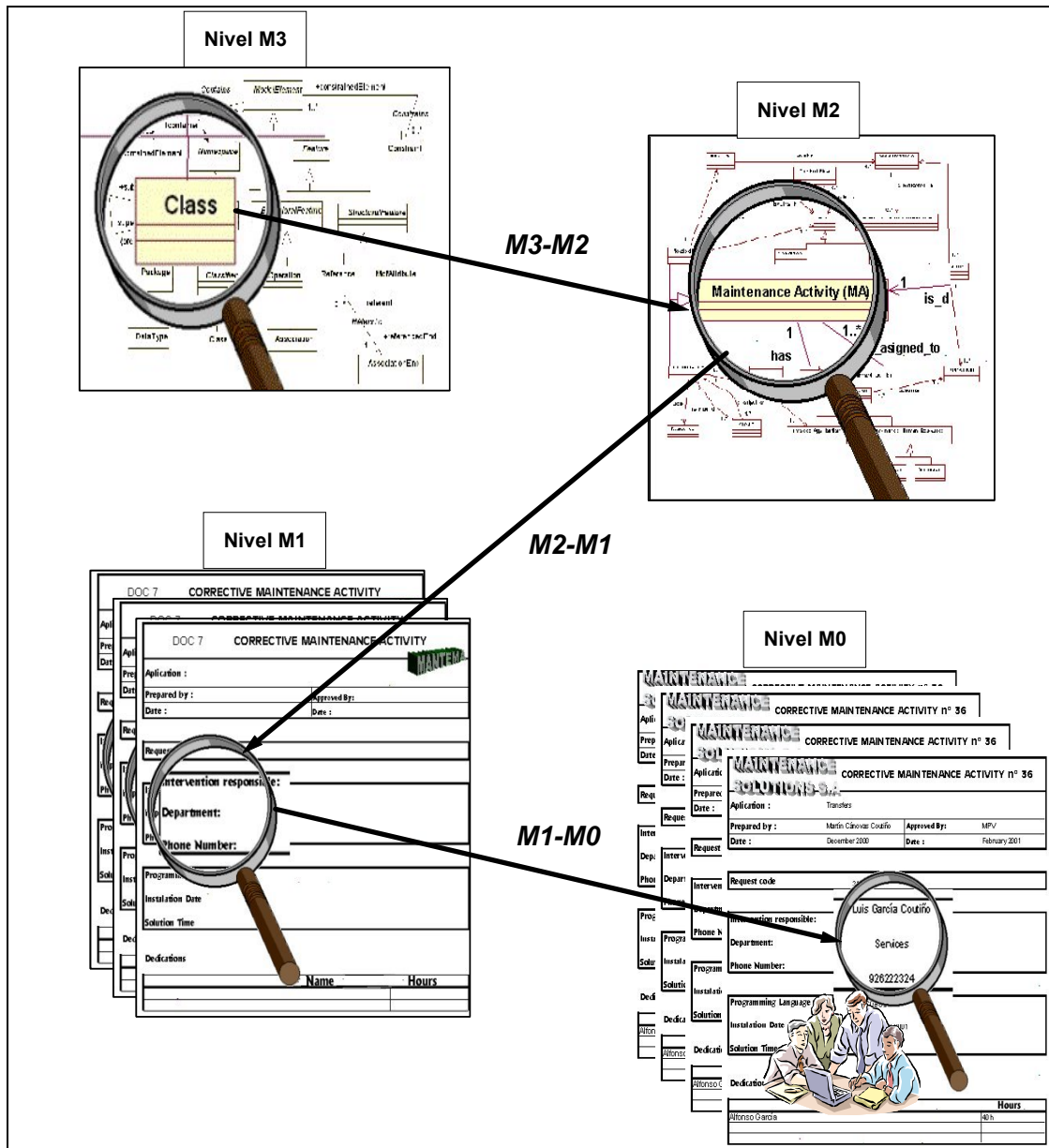


Figura 5-3. Ejemplos de correspondencias entre niveles conceptuales en MANTIS.

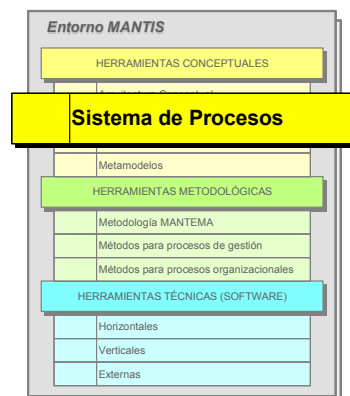
Además de MOF, existen otras propuestas de estándares para arquitecturas conceptuales. Quizás las más conocidas sean la clásica IRDS (*Information Resource Dictionary System*) propuesta por ANSI (1989), o la más reciente OIM (*Open Information Model*) propuesta por la “*Meta-Data Coalition*” (MDC, 1999). En MANTIS hemos optado por MOF por dos razones principales:

- 1) Porque ha sido desarrollada por la OMG, la misma organización que desarrolló UML. Creemos que esto supondrá ventajas futuras de estandarización. De hecho, la nueva versión UML 2.0 que se está desarrollando actualmente, redefine el metamodelo de UML para basarlo en MOF (Duddy, 2002).
- 2) Porque el Modelo MOF del nivel M3 es orientado a objetos y autodefinido lo que supone mayor facilidad de representación, genericidad y flexibilidad que con las otras opciones.

Para una explicación más detallada del uso de esta arquitectura de 4 niveles en MANTIS, se puede consultar el anexo G, en el cual se han desarrollado varios ejemplos de correspondencias M3-M2 y M2-M1 utilizadas en MANTIS.

5.4. Sistema de Procesos.

En el capítulo 3 se han presentado los principales estándares de sistemas de procesos software -según la definición de Wang y King (2000)-. El sistema de procesos del Entorno MANTIS es una adaptación del propuesto en el modelo de referencia ISO 15504-2 (ISO/IEC, 1998c) y su principal utilidad es la sistematización. Los niveles de jerarquía manejados son similares a los que se han definido en el capítulo 3 (de mayor a menor importancia): Sistema, Subsistema, Categoría, Proceso, Grupo (de actividades, también llamado subproceso) y Actividad. En este último nivel se debe tener en cuenta que una actividad puede estar formada por sub-actividades o tareas que, a su vez, pueden estar formadas por otras hasta llegar a actividades atómicas, pero todas estas descomposiciones se engloban en el mismo nivel del sistema de procesos. En base a estos 6 niveles, hemos definido la siguiente taxonomía para el sistema de procesos de MANTIS:



Sistema de Procesos de MANTIS:

- Subsistema Principal.
 - Categoría Principal.
 - Proceso de Mantenimiento.
 - Grupo de Actividades Iniciales Comunes.
 - Grupo de Actividades del Mantenimiento No Planificable.
 - Grupo de Actividades del Mantenimiento Planificable.
 - Grupo de Actividades Finales Comunes.
- Subsistema de Soporte.
 - Categoría de Soporte.
 - Proceso de Gestión de la Configuración.
 - Proceso de Aseguramiento de la Calidad.
 - Proceso de Verificación.
 - Proceso de Validación.
 - Proceso de Revisión Conjunta.
 - Proceso de Auditoría.
 - Proceso de Resolución de Problemas.
- Subsistema Organizacional.
 - Categoría de Gestión.
 - Proceso de Gestión.
 - Proceso de Gestión del Proyecto.
 - Proceso de Gestión de la Calidad.
 - Proceso de Gestión de Riesgos.
 - Categoría de Organización.
 - Proceso de Alineamiento Organizacional.
 - Proceso de Mejora.
 - Proceso de Gestión de Recursos Humanos.

- o Proceso de Infraestructura.
- o Proceso de Medición.
- o Proceso de Reutilización.

La clasificación en subsistemas y categorías de procesos está basada en los estándares ISO 12207 de ciclo de vida (ISO/IEC, 1995), y 15504-2 de modelo de referencia de procesos (ISO/IEC, 1998c), que se han comentado en el capítulo 3. En cuanto a los procesos, no se incluyen todos los referidos en la categoría de soporte en las citadas normas, sino sólo aquellos que tienen interfaz definido dentro de la metodología MANTEMA, es decir, aquellos que son explícitamente invocados desde alguna tarea de dicha metodología (ver anexo B). En cambio, el subsistema organizacional incluye todos los procesos correspondientes definidos en dichas normas, aún cuando el Entorno MANTIS no incluya ninguna herramienta metodológica (capítulo 6) o tecnológica (capítulo 7) para ellos, porque este subsistema de procesos es el básico para el objetivo de gestión del PMS que tiene el Entorno MANTIS. En la Figura 5-4 se muestra un resumen del ámbito que abarca el sistema de procesos de MANTIS respecto de todos los propuestos en la norma ISO 15504-2. Los procesos de soporte (en color amarillo) contribuyen a la gestión del PMS de forma indirecta a través de sus invocaciones desde la metodología MANTEMA. Los procesos organizacionales (en color azul), lo hacen de forma directa mediante las herramientas correspondientes incluidas en el Entorno. En la taxonomía sólo se han indicado los grupos de actividades definidos en la metodología MANTEMA, los cuales son comentados con más detalle en el anexo B.

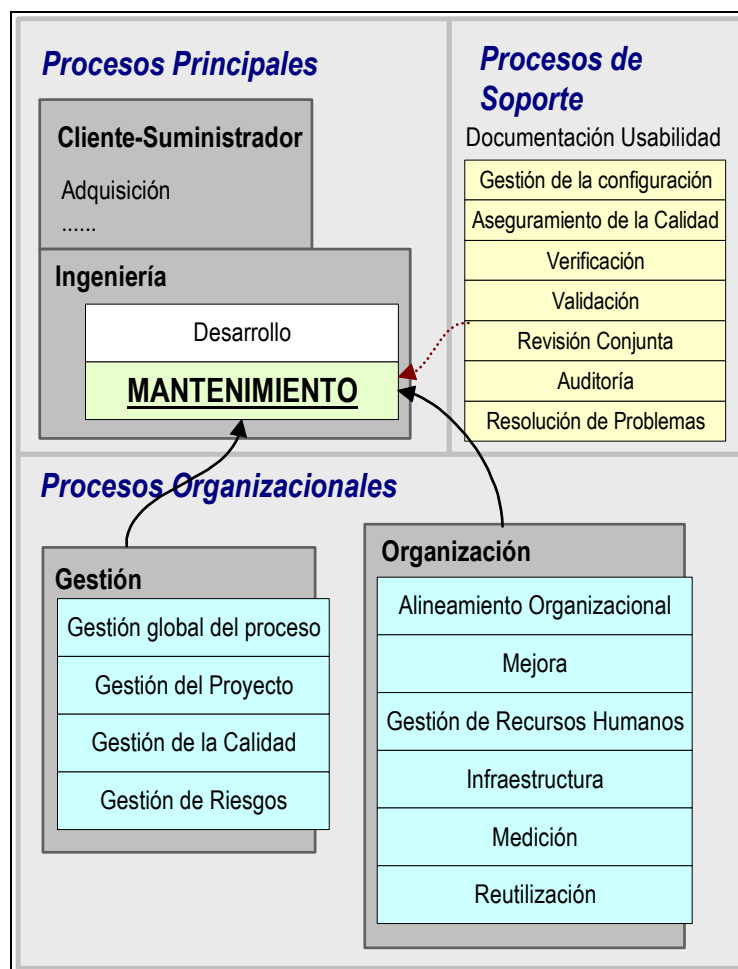


Figura 5-4. Resumen del sistema de procesos de MANTIS.

Todos los niveles de la taxonomía anterior son relativos ya que, en sentido estricto, todos los conceptos asociados (sistema, subsistema, categoría, proceso, grupo de actividades, actividad, subactividad, ...) son de un mismo tipo general, que podríamos llamar “*descripción de trabajo*”, porque satisfacen el mismo patrón “*Basic Process Structure*” de modelado de procesos (Eriksson y Penker, 2000). Las características elementales de este patrón básico (ver Figura 5-5) son las siguientes: La “descripción de trabajo” representa un conjunto de trabajos (procesos, actividades, tareas, ...) que pueden ser realizados. Con ello lo que se pretende es lograr un determinado “objetivo” que provee la motivación para realizar el trabajo. El trabajo utiliza un objeto de una determinada clase como “entrada” para producir otro objeto de otra determinada clase como “salida”. Las clases de la “entrada” y la “salida” pueden ser la misma, en cuyo caso, significa que el trabajo ha consistido en modificar el objeto sin crear otro nuevo. Por último, para realizar el trabajo es necesario utilizar unos determinados recursos.

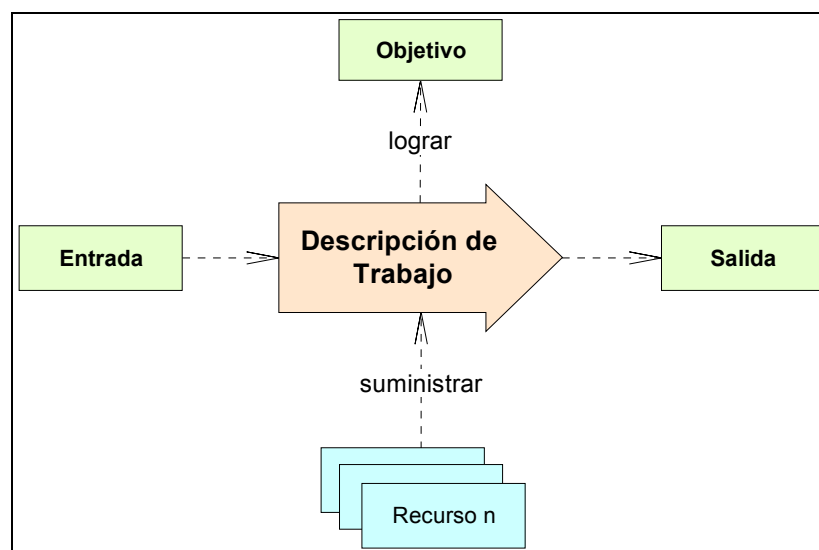


Figura 5-5. Patrón “*Basic Process Structure*”.

En el nivel más bajo de la jerarquía del sistema de procesos de MANTIS está el modelo de actividades y tareas utilizado en la metodología MANTEMA, cuyas principales características son las siguientes:

- Sólo existe un nivel de descomposición de actividades, de manera que cada actividad está formada por una o varias sub-actividades que reciben el nombre de “*tareas*”.
- Cada tarea está caracterizada por las siguientes propiedades:
 - *Código*: Identifica la tarea y que además indica la actividad a la que pertenece y el orden secuencial entre las tareas de la actividad. Por ejemplo, NP2.3 indica la tercera tarea de la actividad NP2 (que a su vez pertenece al grupo NP, es decir, actividades del mantenimiento no planificable).
 - *Nombre*: Es de carácter descriptivo.
 - *Entradas*: Lista de los artefactos (programas, documentos, etc.) utilizados, consumidos o modificados. Se corresponden tanto con el objeto de entrada como con los recursos del patrón de modelado anterior.

- *Salidas*: Lista de los artefactos producidos tras la realización de la tarea. La agregación de todos estos artefactos se corresponde con el objeto de salida del patrón de modelado citado.
- *Técnicas*: Lista de los métodos, técnicas o plantillas útiles.
- *Métricas*: Lista de las métricas que se deben medir durante la realización.
- *Responsables*: Lista de los roles encargados de la realización.
- *Interfaces*: Lista de procesos de soporte que es necesario invocar.

5.4.1. Procesos vs Proyectos.

En el Entorno MANTIS se utilizan los conceptos de proceso y proyecto tal como los definen las normas ISO 12207 (1995) y PMBOK (PMI, 2000). Para comprender mejor el diferente papel de ambos conceptos es útil considerar otra dimensión distinta y ortogonal que viene dada por la dicotomía “datos versus metadatos” (que en MANTIS es resuelta mediante la utilización de la arquitectura conceptual ya comentada). En la Figura 5-6 se representa de forma esquemática esta bi-dimensionalidad, de manera que tanto los procesos como los proyectos pueden analizarse o representarse a nivel de datos o de metadatos.

Los metadatos de procesos son los metamodelos genéricos que permiten abstraer propiedades comunes a grupos o familias de procesos más o menos afines, mientras que los datos se refieren a la representación (es decir, al modelo) de un proceso concreto.

De forma similar, los metadatos de proyectos se refieren a su definición mediante un flujo de trabajo (FT) que representa a una “familia” de proyectos que son especificados y gestionados utilizando los mismos conceptos (la misma ontología), métodos y herramientas, mientras que los datos se refieren a una reificación o instanciación concreta de una definición de un proyecto.

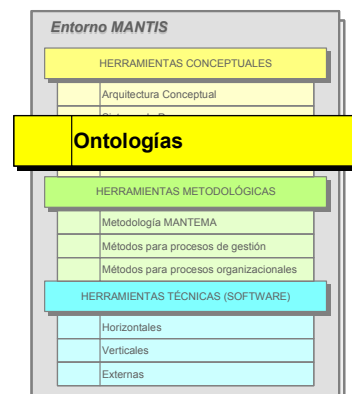
Metadatos	<i>Metamodelos</i> nivel M2	<i>Definición</i> <i>(Flujo de Trabajo)</i> niveles M1 y M2
	<i>Modelos</i> nivel M1	<i>Reificación</i> nivel M0
Datos	Procesos	Proyectos

Figura 5-6. Procesos versus Proyectos.

Cabe resaltar que en los procesos no se considera el nivel de datos (M0) porque son los proyectos los que, en el mundo real, son realizados por personas, consumen recursos, generan nuevos productos o servicios, etc. En cambio, el concepto de proceso es de naturaleza abstracta. Esta es la razón por la que lo que se gestionan (en el mundo real y también en la ontología de MANTIS) son proyectos de mantenimiento de software y no procesos de mantenimiento de software.

5.5. Ontologías.

Para que un EIS sea realmente útil es necesario considerar la integración de todas sus herramientas desde cuatro dimensiones, añadiendo a las tres tradicionales (datos, control e interfaz de usuario) la integración del conocimiento. Sólo considerando adecuadamente esta cuarta dimensión es posible conseguir entornos verdaderamente integrados (Falbo et al, 1999). Para satisfacer este requisito, no basta con permitir diversos niveles de abstracción (utilizando la arquitectura conceptual ya presentada); también es necesario que todos los modelos y metamodelos utilizados en el dominio del problema (la gestión del PMS) estén basados en una misma conceptualización (conjunto de objetos, conceptos, entidades e interrelaciones ente ellos, que se asume que existen en el dominio). Además, es necesario que dicha conceptualización se especifique de manera explícita, es decir, que se construya una ontología según la definición de Gruber (1995).



Por las razones anteriores, un objetivo secundario del Entorno MANTIS ha sido disponer de una ontología común a todos los componentes del Entorno. Por la naturaleza del dominio (la gestión del PMS), esta ontología debe ser bastante amplia, debiendo integrar aspectos pertenecientes a los cuatro tipos de ontologías para sistemas de información definidas por Mylopoulos (1998):

- *Estáticas*, que describen las cosas que existen (normalmente considerando que el sistema estudiado está poblado por “entidades” que tienen identidad única e inmutable), sus atributos y sus interrelaciones.
- *Dinámicas*, que describen los aspectos dinámicos de un sistema en términos de estados, transiciones de estado y procesos.
- *Intencionales*, que abarcan los diversos aspectos relacionados con los agentes utilizando conceptos tales como agente, problema, objetivo, soporte, denegación, sub-objetivo, etc.
- *Sociales*, que abordan el trasfondo social, con las estructuras organizacionales permanentes o las redes cambiantes de alianzas e interdependencias. Tradicionalmente, estas ontologías se han caracterizado en términos de conceptos tales como actor, posición (puesto o responsabilidad), rol, autoridad, compromiso (o acuerdo), etc.

5.5.1. Uso en MANTIS.

Las Ontologías están llamadas a jugar un papel cada vez más importante en la mejora de los procesos de negocio, permitiendo la convergencia de dos puntos de vista hasta ahora divergentes: la ingeniería del software y la gestión de empresas (en particular para el análisis de negocios). Las nuevas tecnologías orientadas al modelado, y en particular las tecnologías de modelado orientado a objetos, suponen una ayuda importante en esta línea (Bertrand y Bezivin, 2000). El uso de la ontología de MANTIS se adapta a estos planteamientos en el sentido de que su principal utilidad no es la implementación de conocimiento sobre el mantenimiento de software (para eso se utilizan los modelos y metamodelos correspondientes a los niveles M1 y M2 de la arquitectura) o sobre proyectos de mantenimiento (para eso se utilizan los datos-objetos del nivel M0 de la arquitectura conceptual), sino ayudar a la compartición de dicho conocimiento entre todos los actores que intervienen en el PMS y, en especial, entre los

ingenieros de mantenimiento y los gestores (de la organización responsable del mantenimiento o de las organizaciones clientes).

Esta primera utilidad de las ontologías en el Entorno MANTIS es evidente, pero esto no hace que deje de ser importante y se le tenga que conceder la atención que como tal problema merece. Su importancia estriba en que la comunicación es una de las principales actividades (en duración y en importancia) en los proyectos de mantenimiento del software (al igual que en cualquier otro tipo de esfuerzo en ingeniería del software) y la ambigüedad del lenguaje natural implica errores, malentendidos y esfuerzos improductivos. Está comprobado que esto es debido en gran parte a que las personas participantes en los proyectos tienen un conocimiento distinto del dominio del problema y/o utilizan un lenguaje diferente, problemas ambos que la disponibilidad de una ontología puede mitigar. Esta es la razón por la que algunos autores han propuesto usar ontologías como espina dorsal de las herramientas y entornos de ingeniería del software (Deridder y Wouters, 1999). Recientemente, Deridder (2002) ha justificado el uso de ontologías como soporte para realizar actividades de mantenimiento ya que en este tipo de problemas la compartición y reutilización de conocimiento (sobre el producto y sus características especialmente) es fundamental.

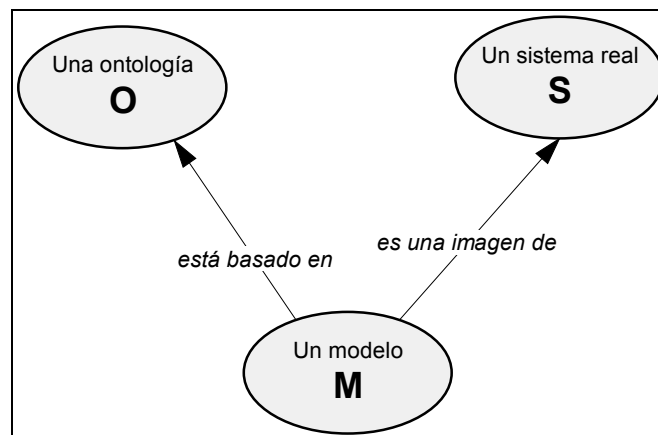


Figura 5-7. La ontología MANTIS es un filtro de conocimiento al definir modelos.

Un segundo uso de la ontología de MANTIS es el filtrado de conocimiento (ver Figura 5-7). Los modelos y metamodelos (modelos de modelos) son representaciones o imágenes de la realidad que, por definición, solo incluyen una parte de dicha realidad. Pero esto no es un problema, sino lo contrario, ya que su utilidad viene dada, precisamente, por su habilidad para filtrar u obviar las características no deseadas. En este sentido, esta ontología también es una ayuda para decidir lo que debe ser extraído de los sistemas reales, en los que se utilice el Entorno MANTIS, para construir los modelos del nivel M1 de la arquitectura conceptual.

En resumen, podemos decir que los usos de las ontologías que se proponen en el entorno MANTIS son dos de los tres identificados por Gruninger y Lee (2002): la comunicación (especialmente entre humanos que participan en proyectos de mantenimiento y entre humanos y el sistema software que también es el Entorno MANTIS), y la reutilización y organización de conocimiento. El tercer uso identificado, la inferencia computacional no ha sido incluido en la propuesta actual de MANTIS, aunque se propone como una de las líneas de trabajo futuras (ver capítulo 8).

Existe una estrecha relación entre las ontologías orientadas a la implementación (también llamadas ontologías de bajo nivel o de aplicación) y los metamodelos, que ya ha sido identificada por algunos autores (Bertrand y Bezivin, 2000). Mientras que un metamodelo busca mejorar el rigor de modelos diferentes pero similares, una ontología hace lo mismo pero para modelos de conocimiento (Devedzic, 2002). Sin una ontología, diferentes representaciones del conocimiento de un mismo dominio pueden ser incompatibles aunque utilicen el mismo modelo de conocimiento.

Así, una ontología formalizada y representada en forma procesable en computadora puede desempeñar un papel similar a un modelo o un metamodelo. Pero en el entorno MANTIS se ha optado por limitar la utilidad de las ontologías a los citados usos de compartición y filtrado de conocimiento y, por ello, no es necesario que las ontologías estén formalizadas completamente. En su lugar, la representación de información en forma computable es realizada mediante los modelos y metamodelos que se comentan en apartados próximos en este mismo capítulo.

5.5.2. Método de Diseño Ontológico.

Diversos autores han postulado la importancia de la formalización ontológica para abordar la definición y diseño de sistemas de información (Guarino, 1998). Por ejemplo, Hikita y Matsumoto (2001) han propuesto utilizar una representación ontológica basada en la lógica de predicados de primer orden para el modelado de procesos de negocio. Pero, para los objetivos de este trabajo no es necesario una formalización plena de las ontologías al estilo de lo anterior, siendo suficiente el nivel de formalización proporcionado por los diagramas de clases de UML, que además tienen la ventaja de su estandarización y extensión, o de otras propuestas textuales semi-formales.

Al igual que pasa con los sistemas software, es recomendable construir las ontologías utilizando un estilo ingenieril, es decir, siguiendo alguna metodología adecuada para tal fin. En MANTIS hemos utilizado el método REFSENO, propuesto por Tautz y Von Wangenheim (1998), que, a su vez, realiza una adaptación y refinamiento de la metodología *Methontology* (Fernández et al, 1997), (Gómez-Pérez, 1998). *Methontology*, imita el ciclo de vida del software propuesto en el estándar IEEE 1074 (IEEE, 1995), estableciendo las etapas principales siguientes:

1. Planificación;
2. Especificación (de los requisitos) de la ontología;
3. Conceptualización, es decir, la ontología propiamente dicha (equivalente al diseño en un sistema software); e
4. Implementación, es decir, representación y almacenamiento de la Conceptualización anterior mediante el uso de alguna herramienta informática.

REFSENO también se basa en estas etapas, aunque nosotros obviaremos la etapa de implementación por que las ontologías no se utilizan con esa finalidad (ver apartado 5.5.1). La especificación de la ontología indica el dominio modelado, el propósito, el alcance e información administrativa como autores y fuentes de conocimiento utilizadas (Blázquez et al, 1998). En la Tabla 5-2 se muestra la especificación de la ontología general del Entorno MANTIS (en adelante simplemente ontología MANTIS).

Concepto	Valor
Dominio	Gestión de Proyectos de Mantenimiento del Software
Autor(es)	Francisco Ruiz, Grupo Alarcos (UCLM)
Propósito	Ontología para permitir el intercambio de información entre los ingenieros, los gestores y los usuarios en proyectos de mantenimiento.
Nivel de formalidad	Semi-formal (diagramas de clases UML y tablas y texto REFSENO).
Alcance	<p><u>Lista de conceptos:</u> Se han clasificado (por razones de claridad) en las siguientes ontologías parciales y sub-ontologías:</p> <ul style="list-style-type: none"> ▪ Ontología del Mantenimiento (proyectos). <ul style="list-style-type: none"> – Sub-ontología de los Productos. – Sub-ontología de las Actividades. – Sub-ontología de Organización del Proceso. <ul style="list-style-type: none"> o Procedimientos. o Gestión de Peticiones. o Problemas. – Sub-ontología de los Agentes. ▪ Ontología de los Flujos de Trabajo. ▪ Ontología de la Medida. <p><u>Lista de instancias:</u> ninguna. <u>Atributos de conceptos comunes:</u> ninguno.</p>
Fuentes de conocimiento utilizadas	Ver Tabla 5-3.

Tabla 5-2. Especificación de requisitos de la ontología de MANTIS.

Aunque en la definición del alcance se recomienda incluir la lista de conceptos, por razones de claridad y extensión, hemos optado por dejar este detalle para la definición de cada una de las ontologías parciales (ver Figura 5-8), por lo que referimos al lector a los glosarios de conceptos de cada una de ellas. En su lugar, en el alcance se indica la estructura básica en que hemos organizado la ontología MANTIS.

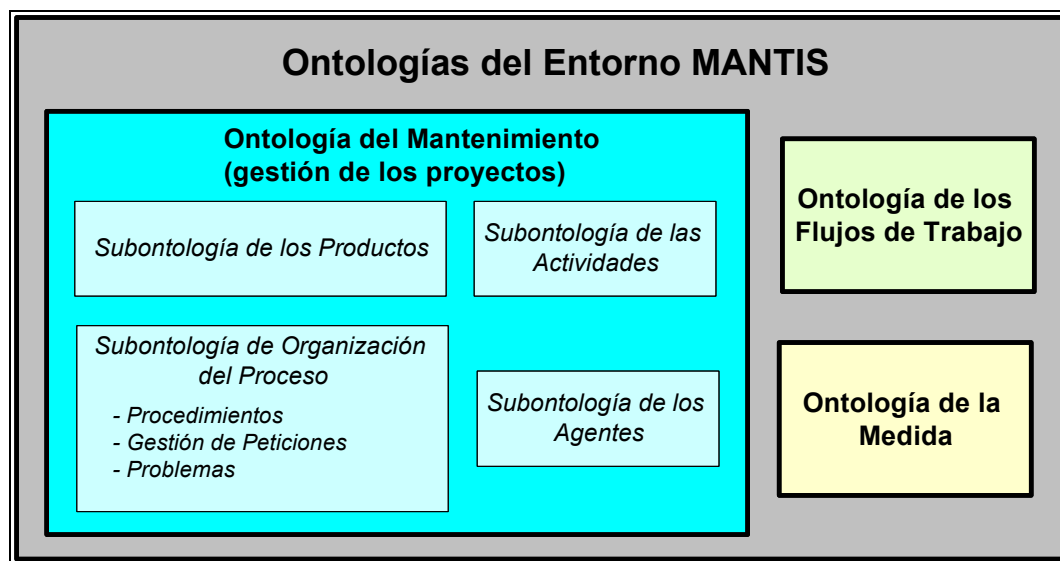


Figura 5-8. Organización de las ontologías de MANTIS.

No se han incluido instancias porque, como ya se ha dicho, no se contemplan los aspectos de implementación. Igualmente no se han definido atributos comunes a varios conceptos por separado sino que, en su caso, se añaden en cada una de las ontologías parciales.

El conocimiento en que nos hemos basado para desarrollar estas ontologías ha sido la experiencia acumulada en el desarrollo de los cuatro proyectos de I+D MANTEMA, MANTICA, MANTIS y MPM (presentados en el capítulo 1), completado con las fuentes documentales incluidas en la Tabla 5-3.

Clave	Fuente Documental
FD1	Propuesta de ontología informal para el PMS de Kitchenham et al (1999).
FD2	Modelo conceptual para el mantenimiento correctivo de Kajko-Mattsson (1999), (2001), (Kajko-Mattsson et al, 2001a).
FD3	Ontología para el proceso de desarrollo de software propuesta por Falbo et al (1998).
FD4	Modelo conceptual de proceso software y de medición del software propuesto por Becker-Kornstaedt y Webby (1999).
FD5	Los documentos que definen el sistema de procesos del Entorno MANTIS presentado en el apartado 5.4:
FD5a	- Modelo de ciclo de vida de ISO 12207 (ISO/IEC, 1995).
FD5b	- Modelo de referencia de procesos ISO15504-2 (ISO/IEC, 1998c).
FD5c	- Propuesta ISO 14764 de modelo del PMS (ISO/IEC, 1998e).
FD5d	- Modelo de actividades y tareas subyacente en la metodología MANTEMA (Polo et al, 1999a).
FD6	Varios modelos de representación de flujos de trabajo:
FD6a	- Modelo de referencia de la <i>Workflow Management Coalition</i> (WfMC, 1995).
FD6b	- Modelo conceptual de Sadiq y Orlowska (1999).
FD7	Varios modelos del proceso de medición:
FD7a	- Propuesta ISO 15939 del proceso de medición (ISO/IEC, 2002).
FD7b	- Modelo conceptual para representar colecciones de datos software propuesto por Kitchenham et al (2001a).

Tabla 5-3. Fuentes documentales utilizadas en la ontología del entorno MANTIS.

La conceptualización o elaboración de la ontología propiamente dicha (segunda etapa del método REFSENO) ha sido realizada, para cada una de las ontologías parciales definidas en el alcance (Tabla 5-2), en los siguientes pasos (Tautz y Von Wangenheim, 1998):

1. Definir el glosario de conceptos a partir de las fuentes de conocimiento citadas.
2. Definir las interrelaciones semánticas entre dichos conceptos representándolas mediante un diagrama de clases UML (esta es una aportación nuestra ya que REFSENO utiliza otro tipo de diagramas parecidos, pero no iguales). A la vez, elaborar una tabla de clases de interrelaciones con las diversas clases que se van identificando.
3. Analizar los conceptos relacionados para identificar las partes comunes a dos o más conceptos. Decidir si estas partes son a su vez conceptos (introducidos por razones de modelado) y, en su caso, incluirlos en el glosario de conceptos.
4. Identificar los atributos terminales (atributos normales) de todos los conceptos e incorporarlos a la(s) tabla(s) de atributos y al diagrama de clases UML. En paralelo, cada vez que se identifica un nuevo tipo de atributo, se debe incluir en la tabla de tipos.

5. Completar las tablas de atributos de conceptos incluyendo los atributos no terminales (equivalentes a las interrelaciones obtenidas en el paso segundo).
6. Comprobar la completitud de todas las tablas de atributos, indicando para cada atributo si pertenece a la capa de descripción del artefacto propiamente dicho, de su interfaz o de su contexto (ver detalles y ejemplos en anexo E).

Una problemática adicional se plantea cuando se integran varias ontologías en una más general. Aunque existen propuestas muy elaboradas que definen procesos complejos muy completos para la integración de ontologías (Pinto y Martins, 2001) (Klein, 2001), en MANTIS hemos optado por una solución más simple consistente en utilizar el método iterativo propuesto por McGuinness et al (2000). De esta manera, las tres ontologías comentadas en este capítulo son el resultado de realizar los tres pasos siguientes durante dos iteraciones:

1. Buscar los lugares (conceptos, atributos o interrelaciones) donde se produce solapamiento.
2. Relatar conceptos que están semánticamente vinculados vía equivalencias o interrelaciones de la misma clase (alineamiento).
3. Chequear la consistencia, coherencia y ausencia de redundancia de los resultados.

5.5.2.1. Sistema de Representación de las Ontologías.

El uso de UML como lenguaje de representación ontológica en ingeniería del software ha sido propuesto recientemente por diversos autores (Wang y Chang, 2001). Además, esta opción está en consonancia con los fundamentos ontológicos de UML según el estudio llevado a cabo por Guizzardi et al (2002). Esta es la razón por la cual, en los apartados próximos, haremos un uso importante de UML para representar gráficamente las ontologías de MANTIS. Otra ventaja del uso de UML es que se pueden utilizar sus posibilidades de extensión (se utilizarán algunas de ellas para los diagramas ontológicos), especialmente mediante estereotipos descriptivos o restrictivos y mediante extensiones, regulares o restrictivas, del metamodelo UML (Schleicher y Westfechtel, 2001).

Además, esta representación gráfica se ha complementado con una representación textual semi-formal basada en el formalismo REFSENO (*Representation Formalism for Software Engineering Ontologies*) elaborado en el Fraunhofer/IESE, el Instituto alemán de Ingeniería del Software (Tautz y Von Wangenheim, 1998). En el anexo E se pueden consultar las descripciones de las primitivas epistemológicas de este formalismo junto con un caso de uso concreto.

Teniendo en cuenta el uso que en el entorno MANTIS se hace de las ontologías (apartado 5.5.1), hemos realizado una adaptación del formalismo REFSENO buscando una mayor sencillez, claridad y compactación de la información. Para ello, nos hemos basado en las siguientes consideraciones:

- a) No considerar las primitivas epistémicas, tablas o columnas de tablas que se refieren a aspectos de la etapa de implementación física de la ontología (por ejemplo, tablas de instancias o funciones de similaridad).
- b) No incluir columnas de tablas que se refieren a aspectos con fines de consulta (por ejemplo, funciones de similaridad).

- c) Utilizar diagramas de clases UML para representar las piezas de información principales de la ontología. En concreto se incluyen los conceptos (el nombre), sus atributos (el nombre) y sus interrelaciones (llamadas atributos no terminales en REFSENO). De estas últimas se indican el nombre, sus roles (cuando están definidos), las cardinalidades máxima y mínima de cada participante y la clase de interrelación (clase del atributo no terminal). Las clases de interrelaciones predefinidas “es-un”, “incluye-a” y “está-compuesto-por” se representan como generalizaciones, agregaciones y composiciones UML, respectivamente, mientras que las demás clases se representan como asociaciones UML indicando su clase como estereotipo de la asociación.
- d) Evitar redundancias de información entre los diagramas y las tablas, usando las tablas sólo para indicar información adicional a la que aparece en los diagramas.

Como consecuencia de estas consideraciones, las primitivas epistémicas de REFSENO, se modifican de la siguiente manera:

- La jerarquía de herencia entre conceptos se incluye en el glosario de conceptos mediante la inclusión de una columna para el super-concepto.
- La colección de tablas de atributos terminales y no terminales de los conceptos se agrupa en únicamente dos tablas. Una llamada tabla de atributos con los terminales y otra, llamada tabla de interrelaciones, con los no terminales ¹⁵.
- En las tablas anteriores se elimina la columna capa porque no aporta información de interés, además, los atributos de la capa de artefacto siempre son terminales y los de la capa de contexto siempre son no terminales, y van a sus respectivas tablas. Los atributos de la capa de interfaz o son de implementación (y entonces no interesan en MANTIS) o no, y entonces van en la tabla de atributo.
- Las tablas de tipos y de clases de interrelaciones (clases de atributos no terminales) son opcionales y sólo se incluyen si se han definido por el diseñador tipos o clases de especial interés en la ontología. Igual ocurre con el glosario de símbolos.

En el anexo E se ha incluido el ejemplo de la sub-ontología de los productos. Si comparamos la representación en REFSENO pura incluida en dicho anexo con la obtenida con nuestra versión para esta misma sub-ontología (apartado 5.5.3.1), se concluye que efectivamente se ha ganado en legibilidad, sencillez y claridad. Los resultados de aplicar este método a las ontologías parciales indicadas en el alcance se presentan en los apartados 5.5.3, 5.5.4 y 5.5.5.

5.5.2.1.1. Ejemplo de Formalización: Concepto de Actividad.

De forma complementaria, y sólo a fines de ejemplo para comprobar que realmente no se aporta más semántica de interés para los objetivos del Entorno MANTIS y se pierde en legibilidad, a continuación incluimos la formalización de algunos de los conceptos definidos en la sub-ontología de las actividades (ver apartado 5.5.3.2). La formalización está realizada en el lenguaje GLEO, que está basado en la lógica de predicados de primer orden (Falbo et al, 1999).

¹⁵ Puesto que los atributos no terminales equivalen a los finales de asociación de las interrelaciones correspondientes, hemos cambiado la nomenclatura porque así creemos que resulta más evidente la semántica subyacente.

La jerarquía de tipos de actividades se formaliza definiendo el predicado $actividad(a,t)$, que significa que ‘ a ’ es una actividad del tipo ‘ t ’, siendo ‘ t ’ alguno de los tipos de actividades incluidos en la taxonomía definida en el sistema de procesos de MANTIS. De esta manera, cada especialización de un tipo de actividad se representa mediante un axioma como el siguiente:

$$(\forall a) (actividad(a, 'Actividad de Modificación') \rightarrow actividad(a, 'Actividad de Mantenimiento'))$$

que representa que una ‘*Actividad de Modificación*’ es una ‘*Actividad de Mantenimiento*’. Adicionalmente, para capturar la descomposición de actividades, se definen los conceptos de sub-actividad y super-actividad, mediante el siguiente axioma:

$$(\forall a1, a2) (sub-actividad(a1, a2) \leftrightarrow super-actividad(a2, a1))$$

Los predicados $entrada(s,a)$ y $salida(s,a)$, junto con $artefacto(s)$ (para indicar que ‘ s ’ es un artefacto según se define en la sub-ontología de los productos), formalizan las relaciones que detallan los artefactos que son entrada o salida de cada actividad:

$$(\forall a, s) (entrada(s, a) \rightarrow artefacto(s) \wedge actividad(a, _))$$

$$(\forall a, s) (salida(s, a) \rightarrow artefacto(s) \wedge actividad(a, _))$$

Las restricciones temporales de precedencia entre dos actividades vienen impuestas por sus entradas y salidas mediante el concepto de pre-actividad (de igual forma se podría definir el concepto simétrico de post-actividad):

$$(\forall a1, a2) (pre-actividad(a1, a2) \leftrightarrow (\exists s) (entrada(s, a2) \wedge salida(s, a1)))$$

El siguiente axioma permite definir formalmente que un artefacto ‘ s ’ se considera salida de una actividad ‘ a ’ que está compuesta por otras sub-actividades ‘ $a1$ ’, ..., ‘ an ’ cuando ‘ s ’ es salida de alguna de sus sub-actividades, sin ser entrada a la vez de alguna de ellas:

$$(\forall a, a1, \dots, an, s) (sub-actividad(a1, a) \wedge \dots \wedge sub-actividad(an, a) \wedge (salida(s, a1) \vee \dots \vee salida(s, an)) \wedge (\neg \exists c) (sub-actividad(c, a) \wedge entrada(s, c)) \rightarrow salida(s, a))$$

Las relaciones de composición y precedencia entre actividades definidas en los axiomas anteriores son transitivas y asimétricas, es decir:

$$(\forall a1, a2, a3) (sub-actividad(a1, a2) \wedge sub-actividad(a2, a3) \rightarrow sub-actividad(a1, a3))$$

$$(\forall a1, a2) (sub-actividad(a1, a2) \rightarrow \neg sub-actividad(a2, a1))$$

$$(\forall a1, a2, a3) (pre-actividad(a1, a2) \wedge pre-actividad(a2, a3) \rightarrow pre-actividad(a1, a3))$$

$$(\forall a1, a2) (pre-actividad(a1, a2) \rightarrow \neg pre-actividad(a2, a1))$$

En resumen, para lo único que podrían interesar estos formalismos dentro del Entorno MANTIS es para representar restricciones (que ni en el sistema de representación de MANTIS ni en REFSENO están contempladas), pero también se podría optar, de forma más coherente quizás, por utilizar el lenguaje de restricciones OCL (*Object Constraint Language*) incluido en UML.

5.5.3. Ontología del Mantenimiento.

Este nombre es una simplificación ya que, realmente, se refiere a la ontología de la gestión de proyectos de mantenimiento de software. Evidentemente, esta ontología contiene la parte central del conocimiento del dominio de interés en el Entorno MANTIS. Las fuentes documentales utilizadas para elaborar esta ontología parcial han sido FD1, FD2, FD3, FD4 y FD5 (según la codificación utilizada en la Tabla 5-3), aunque de todas ellas, la ontología informal para el mantenimiento del software de Kitchenham et al (1999) es la que se utilizó de punto de partida. Aunque dicha propuesta se hizo con el objetivo particular de identificar los factores de contexto que influyen en los resultados de estudios empíricos sobre mantenimiento de software, propone cuatro factores de dominio que influyen sobre el PMS, que nos han servido de base para desglosar la ontología del mantenimiento en cuatro sub-ontologías (por razones de claridad y comprensibilidad). En la Figura 5-9 se muestra un esquema de dichos factores de dominio y los principales conceptos que abarcan. En consecuencia, las sub-ontologías de los productos, de las actividades, de la organización del proceso y de los agentes se corresponden (aunque no al cien por cien) con dichos factores de dominio.

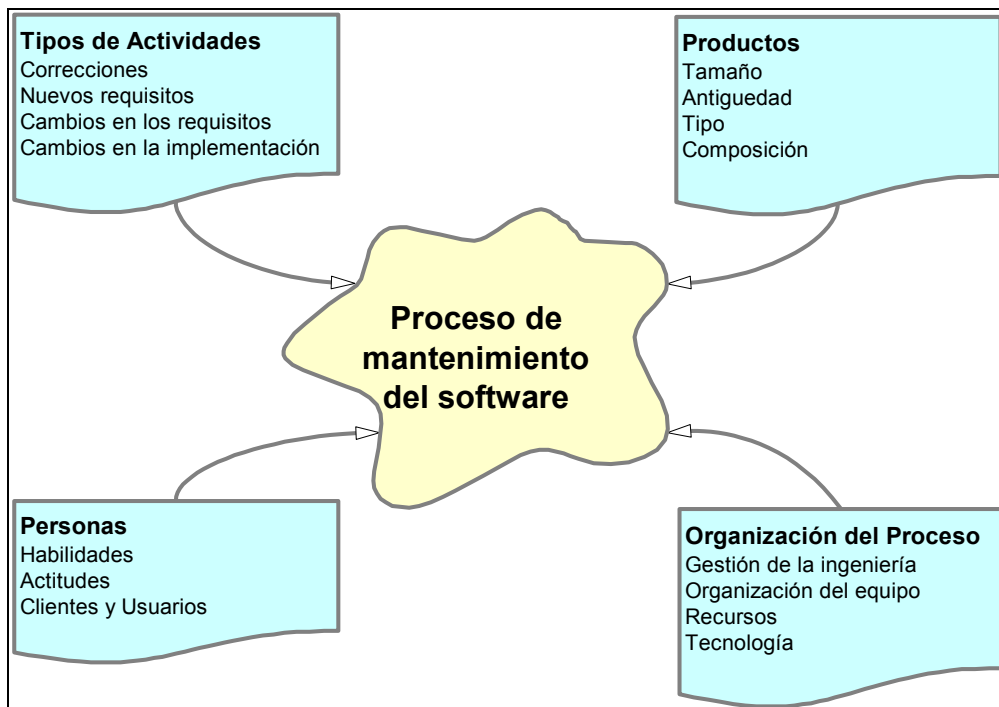


Figura 5-9. Resumen de los factores de dominio que afectan al PMS.

Puesto que el Entorno MANTIS está diseñado para ayudar a la gestión de proyectos reales de mantenimiento de software, la integración de estas cuatro sub-ontologías se realiza por medio del concepto de proyecto tal como lo define la propuesta “*Guide to the Project Management Body of Knowledge (PMBOK Guide)*” del *Project Management Institute* (PMI, 2000). Así, un proyecto es “un esfuerzo temporal emprendido para crear un producto o servicio único”. En esta definición, “temporal” significa que un proyecto siempre tiene un inicio y un final en el tiempo y “único” significa que el producto o servicio es diferente de alguna manera de los demás productos y servicios. Por ejemplo, en el caso del Entorno MANTIS se trata de servicios de mantenimiento que tienen circunstancias específicas en cada proyecto: el software

mantenido, las condiciones de servicio, el cliente y usuarios, etc. Un proyecto se caracteriza, principalmente, porque es:

- Realizado por personas¹⁶;
- Está limitado por restricciones, especialmente en cuanto a recursos.
- Es planificado, ejecutado y controlado (es decir, es gestionado).

Los atributos incluidos en las representaciones ontológicas que siguen son únicamente los que tienen significación especial para la gestión del PMS. Además, sólo incluimos una tabla única con todas las clases de interrelaciones (mostradas como estereotipos en los diagramas UML) para la ontología del mantenimiento. Por esta razón, dicha tabla se incluye en el apartado final 5.5.3.5.

En la Figura 5-10 se muestra el diagrama de clases UML que representa estos conceptos (con el concepto central de proyecto resaltado), junto con las siguientes consideraciones:

- Las restricciones pueden ser precondiciones, que se deben cumplir para poder empezar a ejecutar el proyecto, u objetivos, es decir, post-condiciones que se deben alcanzar para considerar que se ha completado con éxito.
- Un proyecto puede estar formado por varios sub-proyectos.
- En la gestión de un proyecto o sub-proyecto se manejan colecciones de elementos de diversos tipos: artefactos (o productos), actividades, recursos y agentes (humanos o automáticos).

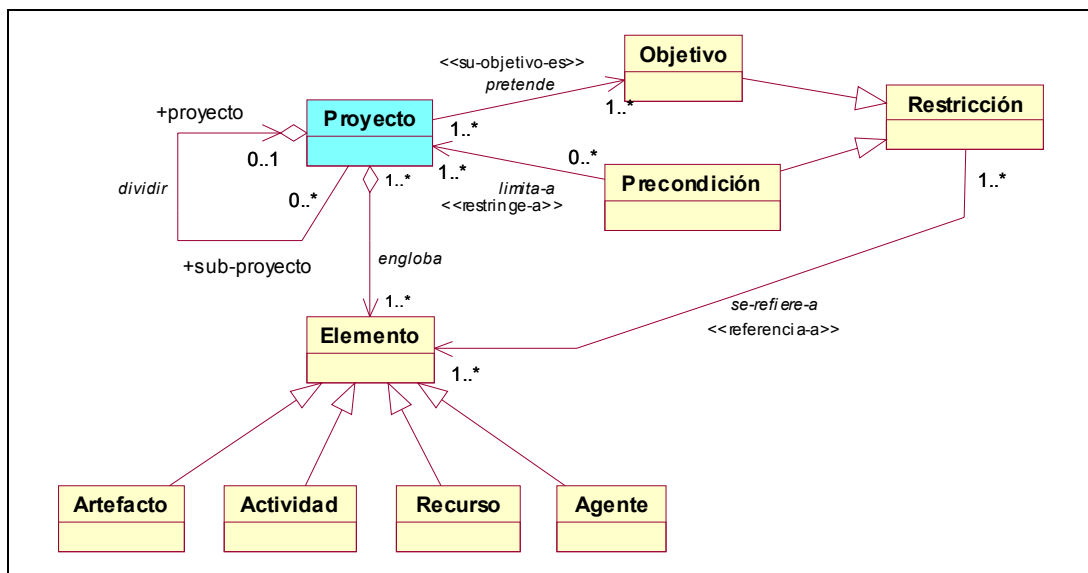


Figura 5-10. Esquema básico de la ontología del mantenimiento.

¹⁶ Con la matización de que se pueden utilizar herramientas que amplían y mejoran las capacidades de dichas personas.

El glosario de conceptos y la tabla de interrelaciones (no mostramos tabla de atributos porque no hay ninguno significativo a este nivel), expresados en la variante del formalismo REFSENO que hemos definido en el apartado 5.5.2.1, son los siguientes:

Concepto	Super-Concepto	Descripción	Propósito
Actividad	-	Ver sub-ontología de las actividades.	-
Agente	-	Ver sub-ontología de los agentes.	-
Artefacto	-	Ver sub-ontología de los productos.	-
Elemento	Concept	Cada uno de los diferentes conceptos manejados en la gestión de proyectos de mantenimiento.	Manejar los diferentes tipos de elementos de un proyecto.
Objetivo	Restricción	Post-condición que debe cumplirse para considerar que un proyecto se ha completado con éxito. Sinónimo: post-condición. Ejemplo: tener construido un producto software concreto.	Gestión de los proyectos.
Precondición	Restricción	Restricción que debe cumplirse antes de que un proyecto puede comenzar a ejecutarse. Ejemplo: Tener designado un jefe de proyecto.	Gestión de los proyectos.
Proyecto	Concept	Esfuerzo temporal emprendido para crear un producto o servicio único (PMI, 2000).	Representar los proyectos de servicios de mantenimiento que una organización lleva a cabo.
Recurso	-	Ver sub-ontología de las actividades.	-
Restricción	Concept	Condición de entorno que limita las opciones posibles para planificar, ejecutar y controlar un proyecto.	Gestión de los proyectos.

Tabla 5-4. Esquema general de la ontología del mantenimiento: glosario de conceptos.

Nombre	Descripción
Dividir	Un proyecto puede dividirse en varios sub-proyectos, que son realizados y gestionados de forma autónoma, aunque coordinada.
Engloba	La gestión de un proyecto de mantenimiento engloba una colección de elementos de diferentes tipos. Un elemento puede aparecer en varios proyectos (una persona participa en más de un proyecto, un recurso se utiliza en más de un proyecto).
Pretende	Un proyecto se lleva a cabo buscando satisfacer unos determinados objetivos.
Limita-a	Un proyecto no puede empezar si no se cumplen unas determinadas precondiciones.
Se-refiere-a	Una restricción se define en referencia a unos determinados elementos de un proyecto. Ejemplo: El personal disponible para el proyecto son el ingeniero “Luis ...” y el programador “Antonio ...”.

Tabla 5-5. Esquema general de la ontología del mantenimiento: tabla de interrelaciones.

5.5.3.1. Sub-ontología de los Productos.

En esta sub-ontología se definen los productos software que son mantenidos, su estructura interna y composición y las versiones que existen de ellos. A continuación se muestran el diagrama UML y las tablas de atributos y de interrelaciones. Aunque el concepto central es el de producto, el concepto de artefacto también es fundamental ya que es uno de los

tipos de elementos que se manejan durante la gestión de proyectos de mantenimiento (Figura 5-10).

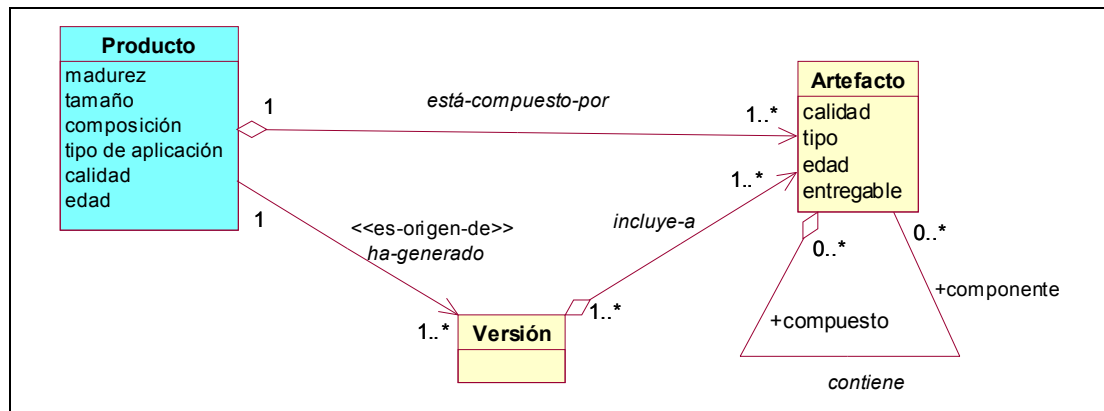


Figura 5-11. Diagrama de la sub-ontología de los productos.

Concepto	Super-Concepto	Descripción	Propósito
Artefacto	Elemento	Parte diferenciada de un producto software que es creada o modificada por las actividades. Puede ser un documento (textual o gráfico), un componente COTS, o un módulo de código. Ejemplos: documento de especificación de requisitos, plan de calidad, módulo de clase, rutina, informe de pruebas, manual de usuario. Sinónimos: elemento software, producto de trabajo, ítem de producto.	Definir la estructura interna y composición del software.
Producto	Concept	Aplicación software que está siendo mantenida. Es un conglomerado de diversos artefactos. Sinónimo: Software.	Mantenerlo.
Versión	Concept	Un cambio en la línea base de un producto. Puede ser una nueva versión completa, <i>upgrade</i> , <i>release</i> o actualización, o un simple parche en el código.	Implantar el proceso de gestión de configuración.

Tabla 5-6. Sub-ontología de los productos: glosario de conceptos.

Concepto	Atributo	Descripción	Cardinalidad
Artefacto	calidad	Medida cualitativa de la calidad, especialmente de la documentación existente sobre el artefacto.	1
	tipo	Naturaleza del artefacto. Ejemplos: documento, módulo, componente, archivo.	1..*
	edad	Número de años desde que se obtuvo la primera versión.	1
	entregable	Indica si el artefacto debe ser entregado al cliente para considerar que el proyecto ha sido completado.	1
Producto	madurez	Etapas en el ciclo de vida del producto según el modelo de Rajlich y Bennett (2000): inicial, evolución, servicio, retirada, o cierre. La frecuencia e importancia de cada tipo de mantenimiento es diferente en cada una de estas etapas (ver capítulo 1).	1
	tamaño	Medida cualitativa del tamaño. Existe una relación clara entre este atributo y la organización y tamaño del equipo de mantenimiento.	1
	composición	Nivel de abstracción de los artefactos que lo forman: componentes tipo caja negra o componentes abiertos con documentos de diseño.	1

Concepto	Atributo	Descripción	Cardinalidad
Producto	tipo de aplicación	La productividad y tipos de mantenimiento están muy influidas por ella. Ejemplos: de gestión, científica, específica, de control, empotrada, etc.	1
	calidad	Medida cualitativa de la calidad, especialmente de la documentación.	1
	edad	Número de años desde que se obtuvo la primera versión. Afecta al mantenimiento en función de la edad del propio producto y también de la edad de la tecnología usada.	1

Tabla 5-7. Sub-ontología de los productos: tabla de atributos.

Nombre	Descripción
Está-compuesto-por	Un producto software está compuesto por artefactos de diferentes clases.
Ha-generado	Un producto software es el origen de una o varias versiones de él a lo largo de su ciclo de vida.
Incluye-a	Una versión incluye un subconjunto de todos los artefactos que forman parte de un producto software.
Contiene	Un artefacto puede estar compuesto por otros más simples y viceversa.

Tabla 5-8. Sub-ontología de los productos: tabla de interrelaciones.

La única interrelación de las cuatro anteriores que no es de una clase predefinida es “ha-generado”. Como se observa en el diagrama UML, su estereotipo indica que es de la clase “es-origen-de”. Por tanto, en la tabla de clases de interrelaciones mostrada en el apartado 5.5.3.5 únicamente es necesario incluir esta clase.

5.5.3.2. Sub-ontología de las Actividades.

Esta sub-ontología engloba dos de los cuatro tipos de elementos básicos para gestionar un proyecto de mantenimiento: actividades y recursos. Más en concreto, en ella se definen los tres aspectos siguientes:

- *Taxonomía de tipos de actividades.* Una actividad es una abstracción de “cómo se hace el trabajo”. Mediante la jerarquía de tipos de actividades se modela el conocimiento de que existen diferentes “tipos de trabajo” a llevar a cabo, siguiendo una correspondencia directa con el sistema de procesos de MANTIS (apartado 5.4).
- *Taxonomía de tipos de recursos.* Los más importantes considerados en el Entorno MANTIS son el hardware y el software, aunque también existen otros como locales, consumibles, etc.). Al contrario que en la propuesta de Kitchenham et al (1999), en MANTIS los recursos son pasivos. Por esta razón, las personas no son un recurso humano (a pesar de ser ésta una denominación muy común) sino agentes que participan activamente en los proyectos. Es importante distinguir entre el concepto de recurso y su materialización. Por ejemplo, un diagrama de clases puede estar materializado en forma de dibujo en papel, en un archivo de texto o en un archivo XML.
- *Relación entre artefactos, actividades y recursos.* Esta relación está basada en el patrón “Basic Process Structure” (ver Figura 5-5), de forma que las actividades consumen (tienen como entrada) y producen o modifican (tienen como salida) artefactos usando unos determinados recursos.

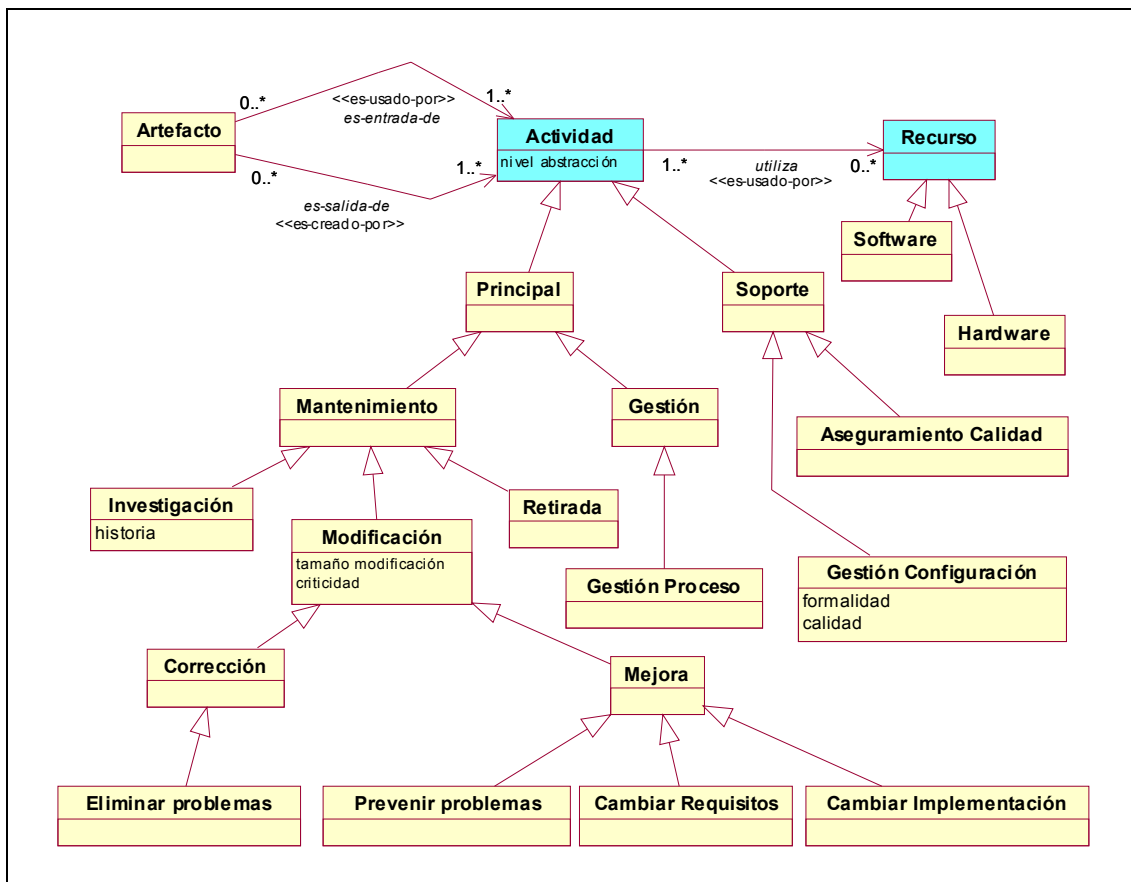


Figura 5-12. Diagrama de la sub-ontología de las actividades.

Es importante tener en cuenta que al referirnos a *Actividades* estamos hablando de cualquier descripción de trabajo entre las presentadas en el sistema de procesos de MANTIS (ver apartado 5.4), es decir, la cantidad de trabajo y complejidad interna puede ser tan grande o pequeña como se desee: una categoría de procesos, un proceso, un grupo de actividades, una actividad, o incluso una sub-actividad (tarea).

En esta sub-ontología no se han incluido todos los tipos de actividades previstos en el sistema de procesos del entorno MANTIS, sino tan sólo aquellos que desempeñan un papel central en la gestión de proyectos de mantenimiento. Así, las actividades se dividen en principales y de soporte. Entre las segundas, las más significativas son las pertenecientes a los procesos de “Aseguramiento de Calidad” y de “Gestión de Configuración”. Las principales se subdividen en actividades propias del proceso de mantenimiento y actividades de gestión de dicho proceso. En estas últimas destacan las actividades de gestión del proceso, que se concretan y detallan en la sub-ontología de organización del proceso (apartado 5.5.3.3.2).

Las actividades de mantenimiento se clasifican, en función del modelo de proceso del estándar ISO 14764 (ISO/IEC, 1998e), en actividades de investigación, de modificación y de retirada. Tal como se explicó en el capítulo 3, las actividades de modificación pueden consistir en corregir los artefactos software o en mejorarlos. Las primeras, que se corresponden con el tipo de mantenimiento correctivo, suelen consistir en la eliminación de problemas detectados en el software. Las segundas pueden ser de tres tipos: prevenir problemas mejorando ciertas características del software (mantenimiento preventivo); implementar cambios en los requisitos o en algunas propiedades de calidad del software (mantenimiento perfectivo); o cambiar sólo

aspectos de implementación, es decir, el entorno operativo del software, sin tocar los requisitos (mantenimiento adaptativo).

No se contempla la descomposición de una actividad en sub-actividades más simples porque la estructura interna se modela de forma más completa y adecuada para la gestión de los proyectos en la ontología de los flujos de trabajo (ver apartado 5.5.4).

El glosario de conceptos y las tablas de atributos e interrelaciones de esta sub-ontología son los siguientes:

Concepto	Super-Concepto	Descripción	Propósito
Actividad	Elemento	Una acción que debe realizarse para lograr los objetivos del proyecto. Sinónimos: Tarea, paso. Ejemplos: análisis de petición de mantenimiento, implementación de la modificación en el código.	Describir el trabajo a realizar.
Artefacto	-	Ver sub-ontología de los productos.	-
Aseguramiento Calidad	Soporte	Actividades de soporte realizadas para asegurar que los productos y los procesos son conformes a los requisitos y a los planes establecidos.	Garantizar la calidad del producto.
Cambiar implementación	Mejora	Actividad de mejora realiza para adaptar un producto software a cambios en su entorno de implementación sin afectar a los requisitos. Sinónimo: Mantenimiento adaptativo. Ejemplo: adaptarlo a un nuevo sistema operativo.	Dar servicio de mantenimiento.
Cambiar requisitos	Mejora	Actividad de mejora realizada para adaptar el funcionamiento de un software a cambios en los requisitos o a la inclusión de nuevos requisitos. Sinónimo: Mantenimiento perfectivo.	Dar servicio de mantenimiento.
Corrección	Modificación	Actividad de modificación que consiste en la eliminación de defectos en un producto software para que su funcionamiento se adapte a los requisitos.	Dar servicio de mantenimiento.
Eliminar problemas	Corrección	Actividad de corrección realizada para eliminar problemas detectados como defectos en el funcionamiento de un software. Sinónimo: Mantenimiento correctivo.	Dar servicio de mantenimiento.
Gestión	Principal	Actividad para gestionar el PMS incluida en el subsistema organizacional definido en el sistema de procesos de MANTIS (apartado 5.4).	Gestionar el PMS.
Gestión Configuración	Soporte	Actividades de soporte cuyo objetivo es establecer y mantener la integridad de los artefactos y hacerlos disponibles, mediante diferentes versiones, a los agentes del proyecto.	Garantizar la integridad de las versiones.
Gestión del Proceso	Gestión	Actividad de gestión realizada para organizar, supervisar y controlar la iniciación y realización del PMS buscando alcanzar sus objetivos.	Gestionar el PMS.
Hardware	Recurso	Recurso formado por un sistema informático, una computadora y un dispositivo periférico.	Utilizar dispositivos informáticos.
Investigación	Mantenimiento	Actividad de mantenimiento que evalúa las diversas maneras de satisfacer una petición de mantenimiento y su impacto en un producto software.	Dar servicio de mantenimiento.

Concepto	Super-Concepto	Descripción	Propósito
Mantenimiento	Principal	Actividad incluida en el PMS según la definición de ISO/IEC (1998e).	Realizar el PMS.
Mejora	Modificación	Actividad de modificación que implementa cambios en un producto software que modifican su comportamiento o implementación o que mejora alguna de sus características de calidad.	Dar servicio de mantenimiento.
Modificación	Mantenimiento	Actividad de mantenimiento que crea o modifica uno o varios artefactos, cambiando el comportamiento o implementación de un producto.	Dar servicio de mantenimiento.
Prevenir problemas	Mejora	Actividad de mejora realizada para eliminar problemas futuros aunque todavía no se han manifestado como defectos. Sinónimo: Mantenimiento preventivo.	Dar servicio de mantenimiento.
Principal	Actividad	Actividad perteneciente al subsistema principal o al subsistema organizacional definidos en el sistema de procesos de MANTIS (apartado 5.4).	Realizar y gestionar el PMS.
Recurso	Elemento	Algo de naturaleza no humana que es necesario para realizar una actividad pero que no forma parte del producto software.	Gestionar recursos.
Retirada	Mantenimiento	Actividad de mantenimiento que se realiza para concluir la vida útil de un producto software.	Concluir un servicio de mantenimiento.
Software	Recurso	Herramienta software utilizada para la automatización total o parcial de alguna(s) actividad(es).	Utilizar herramientas software.
Soporte	Actividad	Actividad cuyo objetivo es facilitar la realización de las actividades principales. Perteneciente al subsistema de soporte definido en el sistema de procesos de MANTIS (apartado 5.4).	Dar soporte a las actividades principales.

Tabla 5-9. Sub-ontología de las actividades: glosario de conceptos.

Concepto	Atributo	Descripción	Cardinalidad
Actividad	Nivel de abstracción	Indica el grado de abstracción de una actividad (declaración de trabajo) dentro de la jerarquía del sistema de procesos de MANTIS (apartado 5.4). Ejemplos: Ciclo de vida, Proceso, Subproceso, Grupo de actividades, Actividad, etc.	1
Gestión de Configuración	Calidad	Grado de cumplimiento de los objetivos del proceso de gestión de configuración. Es un factor determinante para la calidad y eficiencia de un servicio de mantenimiento.	1
	Formalidad	Indicador del nivel de definición del proceso de gestión de configuración. Ayuda a preservar la integridad y consistencia de un producto software y sus artefactos.	1
Investigación	Historia	Información sobre el estado actual y la evolución previa de la actividad de investigación. Su disponibilidad determina en gran parte la eficiencia y calidad de los mantenedores.	1
Modificación	Criticidad	Indica la rapidez con que los usuarios necesitan que la corrección o mejora sea realizada.	1
	Tamaño de la modificación	Estimación del esfuerzo necesario para realizar la modificación, en función de la cantidad de artefactos afectados y su tamaño.	1

Tabla 5-10. Sub-ontología de las actividades: tabla de atributos.

Nombre	Descripción
Es-entrada-de	Las actividades necesitan ciertos artefactos como entradas para poder ser llevadas a cabo.
Es-salida-de	Las actividades producen (crean o modifican) artefactos.
Utiliza	Las actividades necesitan utilizar ciertos recursos para poder ser realizadas.

Tabla 5-11. Sub-ontología de las actividades: tabla de interrelaciones.

5.5.3.3. Sub-ontología de Organización del Proceso.

Esta sub-ontología engloba los conceptos que permiten definir la forma de llevar a cabo las actividades y la forma en que está organizado el proceso de mantenimiento propiamente dicho, incluyendo la definición de las actividades centrales que se realizan durante la planificación y ejecución de un proyecto de mantenimiento.

Una de las principales diferencias entre desarrollo y mantenimiento de software es que el primero está dirigido por los requisitos mientras que el segundo está dirigido por eventos. Esto significa que los estímulos (es decir, las entradas) que inician las actividades de mantenimiento son eventos aleatorios no planificados: la recepción de peticiones de mantenimiento (PM). Por esta razón, es fundamental que una organización de mantenimiento (mantenedor) gestione adecuadamente la cola de peticiones que le llegan. Aunque la metodología MANTEMA (Polo et al, 1999a) incluye una relación detallada de actividades y procedimientos de mantenimiento, en esta sub-ontología hemos generalizado las características comunes a las principales propuestas conocidas (ver capítulo 4), utilizando como eje común el modelo general del PMS propuesto por ISO/IEC (1998e).

Debido a la complejidad del diagrama global de esta sub-ontología (Figura 5-16), hemos optado por dividir su explicación en tres sub-apartados, centrados en los conceptos siguientes:

- *Procedimientos* disponibles para realizar las actividades;
- Actividades de mantenimiento y de gestión que permiten *gestionar las peticiones de mantenimiento* (PM); e
- Identificación de los *problemas* y sus tipos.

5.5.3.3.1. Procedimientos.

Cada actividad puede ser realizada utilizando uno o varios procedimientos, aunque no siempre se utilizan todos ellos porque suelen existir varios procedimientos alternativos para realizar el mismo trabajo (Figura 5-13). Para cada procedimiento se definen su tipo (método, técnica o guía) y las restricciones de utilización que tiene en función de los factores de desarrollo del producto software original: paradigma y tecnología utilizados (Falbo et al, 1998). También se identifican las herramientas software útiles en la automatización de cada procedimiento y los artefactos modificados (o creados) por el procedimiento. Estos últimos son todos o algunos de los definidos como salidas de las actividades asociadas. La tecnología de desarrollo es un factor decisivo a considerar en el caso de productos software con un ciclo de vida previsto muy largo ya que habrá que elegir aquella que tenga garantías de disponibilidad al cabo de muchos años.

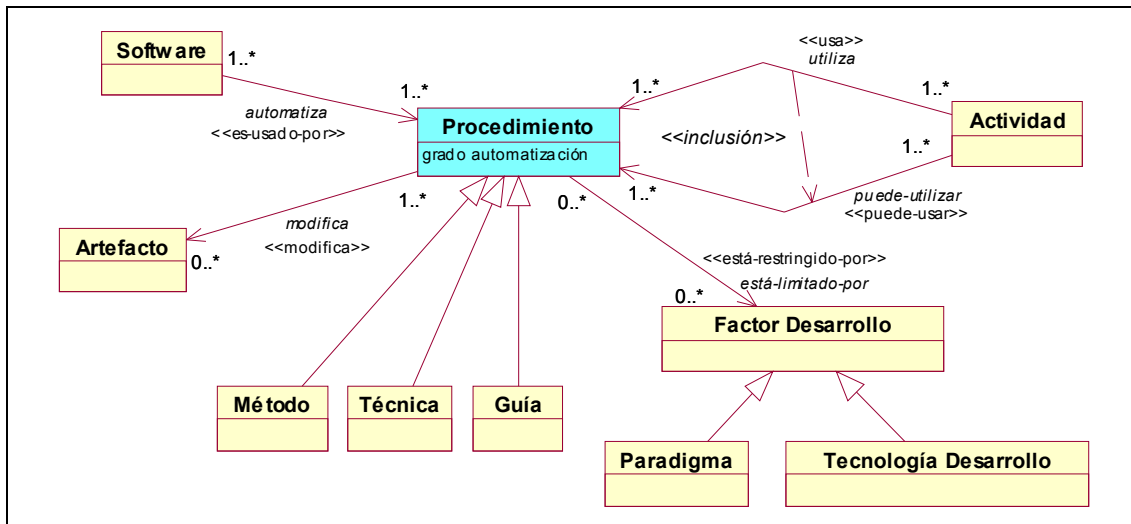


Figura 5-13. Sub-ontología de organización del proceso: los procedimientos.

5.5.3.3.2. Gestión de Peticiones.

En la Figura 5-14 se muestra la parte de la sub-ontología que representa las actividades y artefactos centrales en la gestión del proceso de mantenimiento. La gestión del proceso consiste en una colección de actividades de gestión de peticiones de mantenimiento y de control de cambios (aunque existen otras actividades de gestión del proceso, estas son las fundamentales) que originan actividades de mantenimiento (de investigación y de modificación). Estas actividades utilizan dos clases principales de artefactos: peticiones de mantenimiento (PM) e informes de investigación. El evento que dispara estas actividades es la recepción de una PM. Según su contenido, una PM puede ser de dos tipos diferentes: la descripción de un problema detectado en el software (*informe de problema*) o una petición de un cambio justificada (*solicitud de cambio*). La actividad de “Gestión de PM” recibe la PM y la analiza en función del “Acuerdo de Nivel de Servicio” (ANS) que exista acordado con la organización cliente. En caso de que dicha PM se englobe dentro de las condiciones de servicio del ANS, una actividad de investigación produce un “informe de investigación”. Este informe es recibido por la actividad de “control de cambio” que es la encargada de decidir las actividades de modificación que se aprueban (en caso de que haya alguna). La gestión del proceso también determina la estructura organizativa utilizada por el mantenedor que soporta cada producto software.

Un aspecto complementario en esta parte son las actividades de soporte invocadas por las principales de mantenimiento y de gestión del proceso. En especial, las actividades de “gestión de configuración” (ver capítulo 3 para más detalle) tienen una repercusión muy fuerte sobre la calidad y rendimiento del servicio de mantenimiento, ya que son las encargadas de la entrega a los usuarios de nuevas versiones de un producto software. Además, son básicas para garantizar la integridad y conocer el estado de un producto software y sus artefactos.

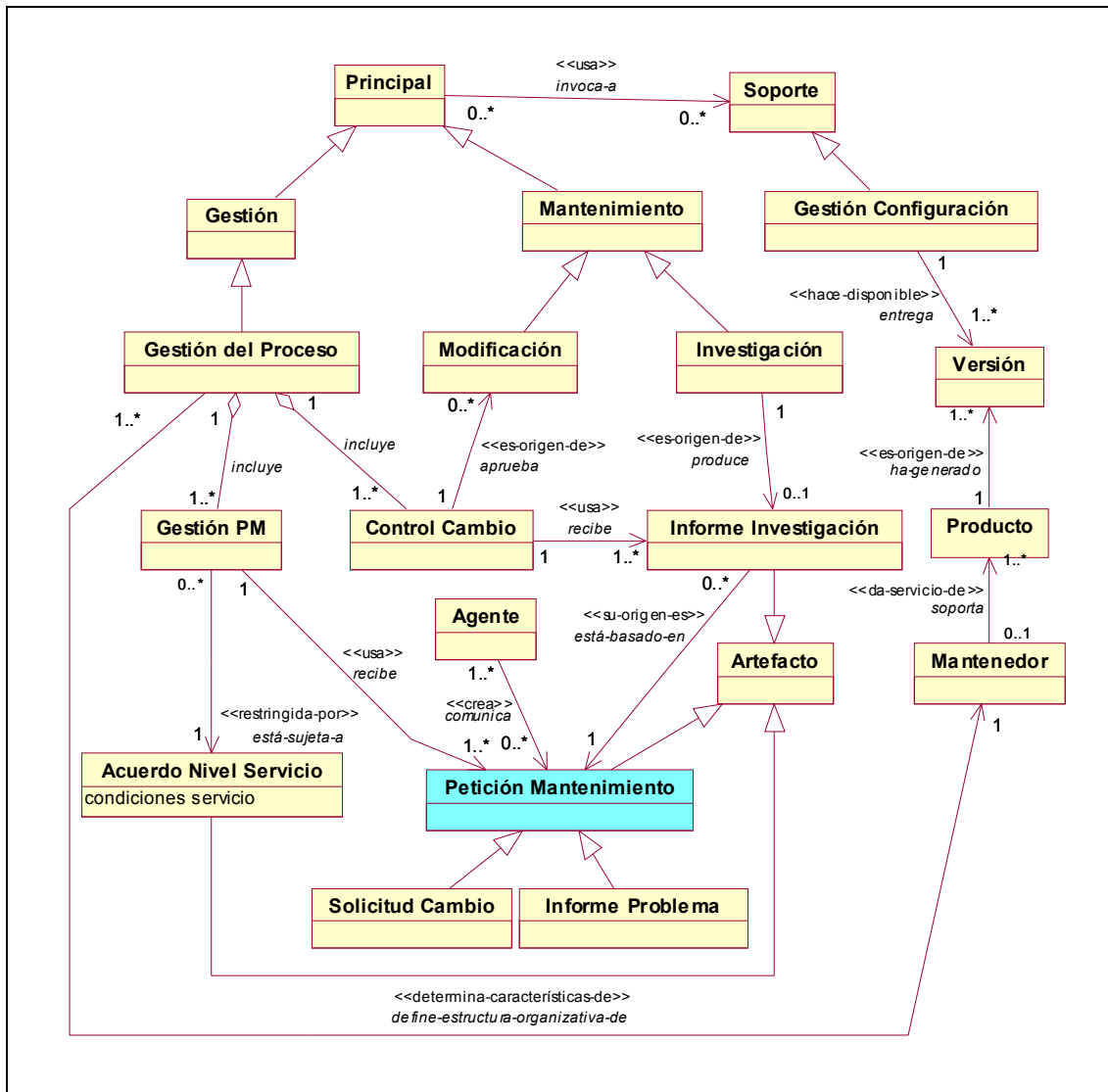


Figura 5-14. Sub-ontología de organización del proceso: gestión de peticiones de mantenimiento.

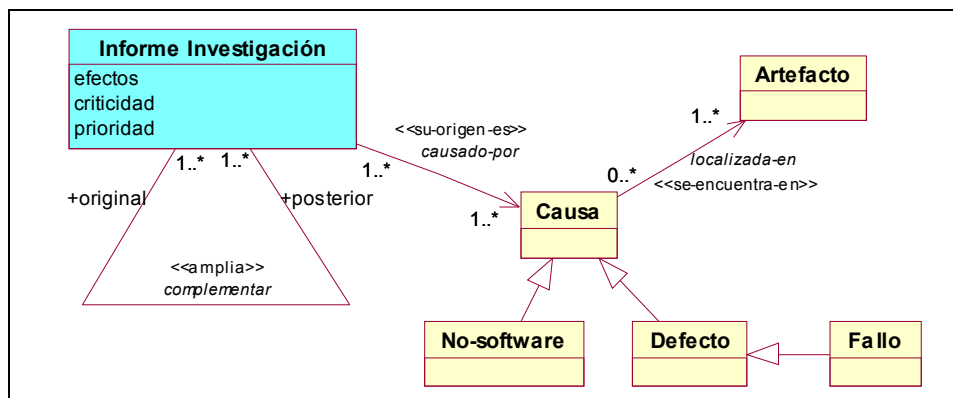


Figura 5-15. Sub-ontología de organización del proceso: informes de investigación.

causas de un problema pueden estar en el propio software (defectos software) o ser de otra naturaleza. Los defectos software más frecuentes son los que producen fallos, es decir, malos funcionamientos durante la ejecución (Kajko-Mattsson, 1999).

5.5.3.3.4. Integración de los Tres Aspectos.

En la Figura 5-16 se muestra el diagrama conjunto integrando los tres aspectos explicados dentro de la sub-ontología de organización del proceso.

El glosario de conceptos y las tablas de atributos e interrelaciones de esta sub-ontología se muestran en las siguientes tablas.

Concepto	Super-Concepto	Descripción	Propósito
Actividad	-	Ver sub-ontología de las actividades	-
Acuerdo Nivel Servicio	Artefacto	Documento con los objetivos y condiciones en que el mantenedor prestará el servicio de mantenimiento al cliente.	Incorporar perspectiva de servicio al PMS.
Agente	-	Ver sub-ontología de los agentes	-
Artefacto	-	Ver sub-ontología de los productos	-
Causa	Concept	Causa de un problema en un producto software.	Analizar los problemas detectados.
Control Cambio	Gestión Proceso	Actividad de gestión que permite determinar las actividades de modificación adecuadas para atender un informe de investigación, y aprobar las modificaciones realizadas.	Autorizar o rechazar modificaciones.
Defecto	Causa	Anomalía de funcionamiento de un producto software respecto a lo establecido en los requisitos.	Clasificar las causas de problemas.
Factor Desarrollo	Concept	Un factor relativo a la manera en que se desarrolló el producto original que supone una restricción para determinar los procedimientos válidos y útiles.	Delimitar el catálogo de procedimientos.
Fallo	Defecto	Defecto encontrado en el código ejecutable.	Clasificar las causas de problemas.
Gestión	-	Ver sub-ontología de las actividades	-
Gestión Configuración	-	Ver sub-ontología de las actividades	-
Gestión PM	Gestión Proceso	Actividad de gestión que consiste en gestionar la cola de PM's, comprobar si se adecuan a las condiciones de servicio establecidas en el ANS y, en su caso, invocar las actividades de mantenimiento adecuadas.	Conocer problemas en el software.
Gestión Proceso	-	Ver sub-ontología de las actividades	-
Guía	Procedimiento	Guía para construir o corregir un tipo específico de documento. Sinónimos: Plantilla, <i>Script</i> .	Definir tipos de documentos a producir.

Concepto	Super-Concepto	Descripción	Propósito
Informe Investigación	Artefacto	Documento que incluye un análisis del impacto de la resolución de un problema o de un cambio solicitado, las diferentes soluciones para implementarlo y la alternativa elegida y su justificación.	Documentar análisis de impacto.
Informe Problema	Petición Mantenimiento	Petición de mantenimiento que describe los síntomas de un problema encontrado en el producto.	Documentar necesidades de mantenimiento.
Investigación	-	Ver sub-ontología de las actividades	-
Mantenedor	-	Ver sub-ontología de los agentes	-
Mantenimiento	-	Ver sub-ontología de las actividades	-
Método	Procedimiento	Procedimiento sistemático, incluyendo sus pasos y heurísticas, para permitir la realización de una o varias actividades. Ejemplo: estimación de esfuerzo mediante COCOMO II.	Definir actividades.
Modificación	-	Ver sub-ontología de las actividades	-
No-software	Causa	Una causa que no es un defecto en el software. Ejemplos: equivocación de los usuarios en la entrada de datos, mal funcionamiento del hardware.	Analizar los problemas detectados.
Paradigma	Factor Desarrollo	Filosofía de desarrollo de software utilizada durante la construcción original del producto. Los procedimientos útiles están limitados por el paradigma. Ejemplo: orientación a objetos.	Identificar procedimientos útiles.
Petición Mantenimiento	Artefacto	Documento describiendo un problema detectado en el producto mantenido (informe de problema) o explicando y justificando un cambio del producto (solicitud de cambio).	Documentar necesidades de mantenimiento.
Principal	-	Ver sub-ontología de las actividades	-
Procedimiento	Concept	Conducta bien definida y precisa para realizar una actividad. Sinónimo: directiva.	Explicar la manera de realizar las actividades.
Producto	-	Ver sub-ontología de los productos	-
Software	-	Ver sub-ontología de las actividades	-
Solicitud Cambio	Petición Mantenimiento	Petición de mantenimiento que explica y justifica la necesidad de un cambio en el producto.	Documentar necesidades de mantenimiento.
Soporte	-	Ver sub-ontología de las actividades	-
Técnica	Procedimiento	Procedimiento usado para realizar una actividad definido de forma menos rigurosa que un método. Técnica: estimación funcional de tamaño.	Definir actividades.
Tecnología Desarrollo	Factor Desarrollo	La tecnología utilizada cuando el producto y sus artefactos fueron originalmente desarrollados. En función de cual esta tecnología los procedimientos de mantenimiento podrán ser unos u otros. Ejemplos: almacén de datos, SGBD relacional.	Decidir procedimientos válidos.
Versión	-	Ver sub-ontología de los productos	-

Tabla 5-12. Sub-ontología de organización del proceso: glosario de conceptos.

Concepto	Atributo	Descripción	Cardinalidad
Acuerdo Nivel Servicio	Condiciones servicio	Descripción del alcance del servicio de mantenimiento, es decir, de los compromisos que adquiere el mantenedor frente al cliente.	1..*
Informe Investigación	Efectos	Descripción de los efectos producidos por un problema concreto en el funcionamiento del software.	1..*
	Criticidad	Evaluación cualitativa del grado de impacto que la resolución del problema tiene en el funcionamiento ordinario del producto.	1
	Prioridad	Nivel de importancia asignado a la resolución del problema. Está determinado en parte por la criticidad.	1
Procedimiento	Grado automatización	Indicador del nivel de automatización actual de un procedimiento: ninguno, semi-automático, automático.	1

Tabla 5-13. Sub-ontología de organización del proceso: tabla de atributos.

Nombre	Descripción
Aprueba	El resultado de la actividades de control de cambios es la aprobación de modificaciones.
Automatiza	Cada recurso software automatiza unos determinados procedimientos.
Causado-por	El problema analizado en un informe de investigación está originado por una o varias causas.
Complementar	Un informe de investigación puede complementar a otros anteriores.
Comunica	Los agentes que participan en el proyecto envían las PM's al mantenedor.
Define-estructura-organizativa-de	En las actividades de gestión del proceso se define la estructura organizativa utilizada por el mantenedor para llevar a cabo el proyecto.
Entrega	Las actividades de gestión de configuración entregan nuevas versiones y mejoras.
Está-basado-en	Cada informe de investigación está basado en una petición de mantenimiento.
Está-limitado-por	La utilidad de un procedimiento está limitada por los factores de desarrollo del producto software.
Está-sujeta-a	Las decisiones en las actividades de gestión de PM están sujetas a lo estipulado en el ANS.
Ha-generado	Ver sub-ontología de los productos.
Incluye	La gestión del proceso incluye, entre otras, un conjunto de actividades de gestión de PM y de control de cambio.
Invoca-a	Las actividades principales invocan a actividades de soporte.
Localizada-en	Cada causa de un problema se encuentra localizada en uno o varios artefactos.
Modifica	Cada procedimiento puede modificar (o crear) unos determinados artefactos.
Produce	Las actividades de investigación producen informes de investigación.
Puede-utilizar	Para realizar cada actividad se pueden utilizar unos determinados procedimientos.
Recibe	Las actividades de gestión de PM se activan cuando se recibe una petición de mantenimiento.
Soporta	Ver sub-ontología de los agentes.
Utiliza	En cada actividad se utilizan algunos de los procedimientos que son utilizables para su realización.

Tabla 5-14. Sub-ontología de organización del proceso: tabla de interrelaciones.

5.5.3.4. Sub-ontología de los Agentes.

Esta sub-ontología se refiere a la *jerarquía de tipos agentes* que existen durante la gestión de proyectos de mantenimiento. Esta jerarquía tiene las características siguientes:

- Los agentes pueden ser humanos o automáticos (herramientas software).
- Los agentes humanos pueden ser personas individuales u organizaciones. Estas últimas están formadas por personas, de manera que cada persona desempeña un determinado puesto en cada organización que la emplea (Becker-Kornstaedt y Webby, 1999).
- Cada organización tiene un modelo organizacional, que se representa mediante una jerarquía de agregación de organizaciones subordinadas completado con las relaciones de subordinación entre las personas que forman parte de ellas (en realidad, entre los puestos que ocupan).
- Existen tres tipos de organizaciones virtuales: mantenedor, cliente y usuario (según el modelo de organizaciones participantes definido en la metodología MANTEMA, ver anexo B). Se identifican como virtuales porque pueden corresponder a una organización real diferente cada una de ellas o las tres ser la misma organización real. También es común el caso en que existen dos organizaciones reales, el mantenedor y el cliente, que a la vez es usuario. También puede ocurrir que las tres organizaciones virtuales sean organizaciones reales subordinadas de una misma organización real, por ejemplo, cuando un departamento de informática da servicio de mantenimiento a otros departamentos de la misma empresa.

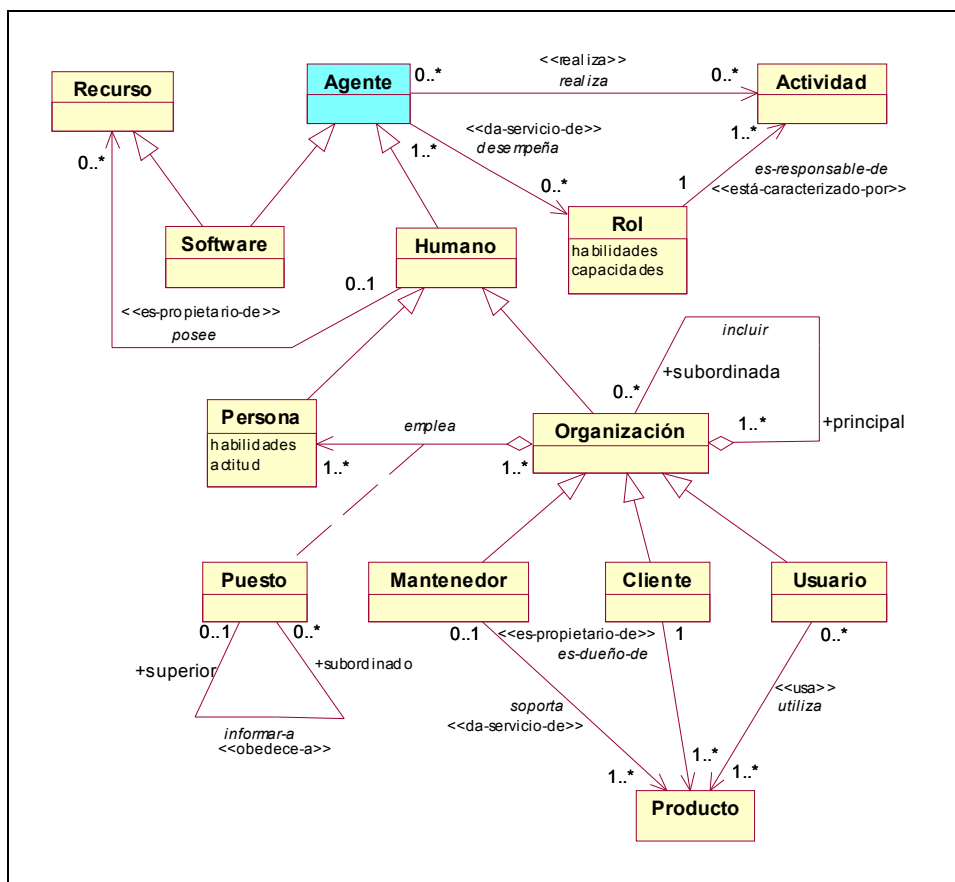


Figura 5-17. Diagrama de la sub-ontología de los agentes.

Como muestra la Figura 5-17, el modelo de agentes también está basado en el concepto de rol, lo que permite mayor generalidad y flexibilidad para representar cualquier posible situación que se produzca en un proyecto real (Polo et al, 1999b). De esta manera, durante la definición de los procesos, se establecen los roles existentes y las actividades que son responsabilidad de cada rol. En cambio, durante la planificación del proyecto se establecen los agentes participantes y los roles que desempeñarán. Por último, durante la reificación de los procesos, es decir, durante la ejecución del proyecto, se identifican las realizaciones concretas de actividades de cada agente.

El glosario de conceptos y las tablas de atributos e interrelaciones de esta sub-ontología son los siguientes:

Concepto	Super-Concepto	Descripción	Propósito
Actividad	-	Ver sub-ontología de las actividades	-
Agente	Elemento	Una persona, organización o aplicación software que desempeña un papel activo en la realización de un proyecto de mantenimiento, es decir, realiza alguna(s) actividad(es). Sinónimo: actor.	Identificar los participantes en el proyecto.
Cliente	Organización	Organización propietaria del producto software mantenido.	Definir las organizaciones participantes.
Humano	Agente	Agente que es una persona individual o un grupo de personas (organización).	Definir estructural organizacional.
Mantenedor	Organización	Organización que realiza el servicio de mantenimiento del producto software.	Definir las organizaciones participantes.
Organización	Humano	Un agente humano formado por un grupo de personas u organizaciones más simples (sub-organizaciones) que actúan con una identidad común en el proyecto. Ejemplos: empresa cliente, equipo de mantenimiento, departamento de informática.	Definir estructural organizacional.
Persona	Humano	Agente humano individual.	Identificar personas participantes.
Producto	-	Ver sub-ontología de los productos	-
Puesto	Concept	Un puesto de trabajo dentro de una organización. Es ocupado por una persona. Ejemplos: Jefe de proyecto, Director de TI.	Definir estructural organizacional.
Recurso	-	Ver sub-ontología de las actividades	-
Rol	Concept	Abstracción de un conjunto de habilidades o capacidades necesarias para realizar una o varias actividades. Sinónimos: papel, función. Ejemplos: responsable de pruebas, ingeniero de mantenimiento.	Establecer responsabilidades.
Software	-	Ver sub-ontología de las actividades	-
Usuario	Organización	Organización que utiliza el producto software mantenido.	Definir las organizaciones participantes.

Tabla 5-15. Sub-ontología de organización del proceso: glosario de conceptos.

Concepto	Atributo	Descripción	Cardinalidad
Persona	Actitud	La actitud o motivación de una persona para realizar su trabajo. Produce un impacto importante en la calidad de los procesos.	1
	Habilidades	Habilidades que posee una persona (relacionadas con proyectos software). Similares a las de los roles.	1..*
Rol	Capacidades	Capacidades (conocimientos y dominios) necesarias para poder desempeñar un rol. Ejemplo: análisis y diseño con UML.	1..*
	Habilidades	Habilidades personales que ayudan a poder desempeñar mejor un rol. Ejemplo: liderazgo.	1..*

Tabla 5-16. Sub-ontología de organización del proceso: tabla de atributos.

Nombre	Descripción
Desempeña	Un agente puede desempeñar varios roles en un proyecto. Igualmente, un rol puede ser desempeñado por varios agentes diferentes.
Emplea	Cada persona en cada organización es empleada en un determinado puesto.
Es-dueño	Un recurso puede tener un propietario, que debe ser un agente humano.
Es-responsable-de	Cada actividad tiene un rol que es responsable de su correcta realización.
Incluir	Las organizaciones pueden estar formadas por otras organizaciones más simples (sub-organizaciones) y pueden ser parte de otras organizaciones más complejas.
Informar-a	Los puestos de trabajo de una organización tienen una organización jerárquica de forma que cada puesto informa y obedece a su inmediato superior.
Posee	Los agentes humanos pueden ser propietarios de los recursos utilizados en el proyecto.
Realiza	Durante la fase de ejecución del proyecto, agentes concretos realizan las actividades.
Soporta	La organización de mantenimiento (mantenedor) da servicio de mantenimiento de unos determinados productos software.
Utiliza	Una organización usuario utiliza unos determinados productos software.

Tabla 5-17. Sub-ontología de las actividades: tabla de interrelaciones.

5.5.3.5. Aspectos Comunes.

La integración de estas cuatro sub-ontologías se ha realizado teniendo en cuenta lo comentado sobre ingeniería ontológica (apartado 5.5.1). Los diagramas y tablas anteriores ya están libres de inconsistencias semánticas y sintácticas. La única excepción es en el caso de repeticiones de nombres de interrelaciones ya que, en el caso de ontologías tan complejas como las presentes, es prácticamente imposible evitar esta duplicidad si se desea utilizar nombres auto-explicativos. Esto no es problema siempre y cuando se cumpla que las dos interrelaciones etiquetadas con igual nombre tengan la misma clase de interrelación (es decir, su semántica sea la misma) y que, en caso de duda, se precedan o acaben con el nombre de un concepto participante que permita distinguirlas; por ejemplo, [Agente].posee o incluye.[Gestión PM].

La tabla de clases de interrelaciones de la ontología del mantenimiento (englobando las cuatro sub-ontologías anteriores), es la siguiente:

Clase	Nombre inverso	Propósito
Amplía	Es-ampliado-por	Las instancias de un concepto permiten conocer en más detalle las características de las instancias de otro concepto.
Caracteriza-a	Está-caracterizado-por	Una instancia de un concepto identifica a las instancias de otro concepto.
Crea	Es-creado-por	Una instancia de un concepto provoca la creación de instancias de otro concepto.
Da-servicio-de	Es-servido-por	Las instancias de un concepto llevan a cabo un trabajo para atender una necesidad de las instancias de otro concepto.
Determina-características-de	Sus-características-están-determinadas-por	La estructura interna de las instancias de un concepto es determinada por las instancias de otro concepto.
Es-objetivo-de	Su-objetivo-es	La existencia de una instancia de un concepto es la razón de la existencia de instancias de otro concepto.
Es-origen-de	Su-origen-es	Una instancia de un concepto se origina como consecuencia y después de la existencia de otra instancia de otro concepto.
Es-propietario-de	Su-propietario-es	Las instancias de un concepto que refiere a personas u organizaciones humanas, son propietarias de instancias de otro concepto.
Hace-disponible	Está-disponible-por	La existencia de una instancia de un concepto hace posible que las instancias de otro concepto pueden ser utilizadas por instancias de otros conceptos diferentes.
Modifica	Es-modificado-por	Una instancia de un concepto provoca la modificación de instancias de otro concepto.
Obedece-a	Es-obedecido-por	Las acciones de las instancias de un concepto están determinadas por instancias de otro concepto.
Puede-usar	Es-usable-por	Una instancia de un concepto puede utilizar –opcionalmente- una instancia de otro concepto.
Realiza	Es-realizado-por	Las instancias de un concepto que refiere a un trabajo o acción, son realizadas por instancias de otro concepto.
Referencia-a	Referenciado-por	La definición de una instancia de un concepto se hace en referencia a instancias de otro concepto.
Restringe-a	Restringido-por	Los estados posibles de una instancia de un concepto están limitados por instancias de otro concepto.
Se-encuentra-en	Es-encontrada-en	Las instancias de un concepto están localizadas dentro de la estructura interna de instancias de otro concepto.
Usa	Es-usado-por	Una instancia de un concepto utiliza instancias de otro concepto.

Tabla 5-18. Ontología del mantenimiento: tabla de clases de interrelaciones.

5.5.4. Ontología de los Flujos de Trabajo.

Esta ontología ha sido realizada para ampliar la ontología del mantenimiento, ya comentada, con tres aspectos importantes para la gestión de proyectos de mantenimiento (Ruiz et al, 2001a), (Ruiz et al, 2001b):

- Descomposición de una actividad compleja en actividades más simples.
- Restricciones temporales entre unas y otras actividades, es decir, el orden de realización de las actividades.
- Control del estado de realización de las actividades y de los proyectos durante la reificación de los procesos.

Aunque el primer aspecto ya se ha modelado en la sub-ontología de las actividades mediante la agregación “componer” (ver Figura 5-12), para poder representarlo de forma integrada con el segundo (temporalidad), hemos optado por basarnos en el concepto de “*Flujo de Trabajo*” (FT), también conocido por su nombre inglés “*WorkFlow*”. Esta decisión está en consonancia con diversas propuestas, formuladas en los últimos años, que abogan por la utilidad de la tecnología de FT’s en el ámbito de los entornos de ingeniería del software.

En este sentido, Ocampo y Botella (1998) han sugerido la posibilidad de utilizar FT’s para modelar y reificar los procesos software, sacando partido de la similitud existente entre ambas tecnologías¹⁷. Este planteamiento ha sido confirmado por algunos desarrollos concretos de herramientas basadas en FT para dar soporte al proceso software en general (Penadés et al, 1999a y 1999b). Por otro lado, la utilidad de los “*Sistemas de Gestión de Flujos de Trabajo*” (SGFT) en la automatización de procesos de negocio ha sido demostrada con creces y, puesto que en MANTIS el PMS es considerado con una perspectiva de proceso de negocio, parece razonable considerar que la tecnología de FT es capaz de aportar al PMS la automatización de esta perspectiva más amplia, en línea con los objetivos del Entorno MANTIS (ver capítulo 7 para más detalle). Este planteamiento ha sido verificado en casos reales consistentes en la utilización de un SGFT para la gestión del PMS por Aversano et al (2001).

En cuanto al tercer aspecto contemplado en esta ontología, también nos hemos basado en la tecnología de FT’s. En concreto, hemos utilizado el modelo de transición de estados definido por la *WfMC* (1999) para definir el concepto de estado de una actividad.

En consonancia con los comentarios anteriores, la ontología de los flujos de trabajo del Entorno MANTIS está basada en dos fuentes documentales (códigos FD6a y FD6b en Tabla 5-3) provenientes del ámbito de la tecnología de FT:

- El modelo de referencia propuesto por la *WfMC* (1995)¹⁸; y
- El modelo conceptual de FT’s propuesto por Sadiq y Orlowska (1999).

Antes de pasar a representar la ontología de los flujos de trabajo propiamente dicha, incluimos dos apartados describiendo los modelos conceptuales utilizados como base para su elaboración.

5.5.4.1. Modelo Conceptual de los Flujos de Trabajo.

Tradicionalmente, un FT se ha representado utilizando grafos acíclicos dirigidos (GAD) como el de la Figura 5-18, que esquematiza el modelo de FT’s propuesto por Sadiq y Orlowska (1999). De esta forma, un FT está formado por un conjunto de “Nodos” (figuras geométricas)

¹⁷ En ambos casos (procesos software y procesos de negocio) se habla de actividades que forman el proceso, de artefactos (módulos, documentos, etc.) que modifican o producen dichas actividades, de agentes que participan desempeñando ciertos roles, de recursos utilizados (por ejemplo, aplicaciones software), de aspectos organizacionales (colaboración en equipos de trabajo), etc.

¹⁸ OMG también ha adoptado este modelo de referencia. Prueba de esta colaboración ha sido la propuesta “*Workflow Management Facility*” (OMG, 2000), que define interfaces IDL para el control y supervisión de la ejecución de FT’s, y para la interoperatividad entre SGFT’s, y utiliza una notación gráfica basada en diagramas de interacción UML.

interconectados mediante "Flujos de control" (flechas). Los nodos pueden ser "Actividades", representadas por rectángulos en la figura, o "Condiciones de Transición", representadas por círculos. Una condición puede ser de tipo "*Or-split*" o de tipo "*Or-join*". Las condiciones Or-split y Or-join permiten representar bifurcaciones y fusiones, es decir, caminos de ejecución opcionales. Para representar caminos de ejecución concurrentes se utilizan actividades con más de un flujo de control empezando en ellas (inicio de la concurrencia, también llamado "*And-split*") o más de un flujo de control acabando en ellas (final de la concurrencia o sincronización, también llamado "*And-join*"). Estos cuatro tipos de condiciones de transición coinciden con los establecidos en el modelo de referencia de la WfMC (1995).

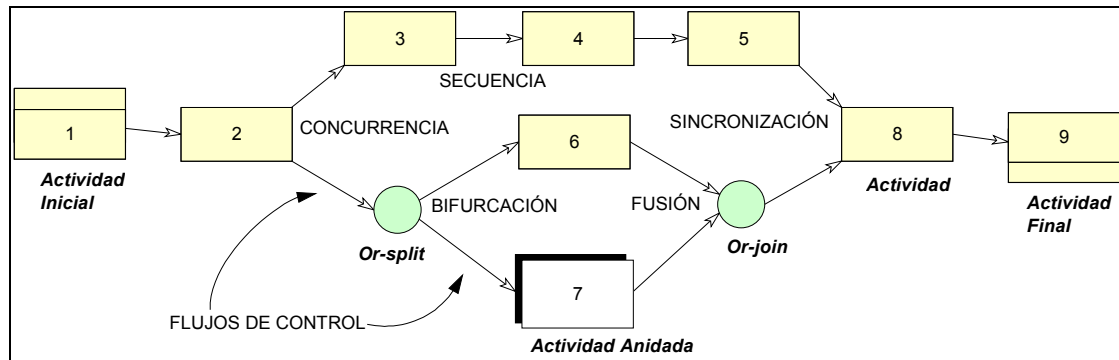


Figura 5-18. Ejemplo de diagrama de flujo de trabajo.

Combinando nodos y flujos de control es posible obtener los cinco tipos de estructuras representadas en la figura (secuencia, concurrencia, sincronización, bifurcación y fusión). Adicionalmente, para una correcta estructuración del grafo (Sadiq y Orłowska, 1996), se debe cumplir la condición de que sólo haya un único nodo sin flujos de control entrantes (actividad inicial) y uno sólo sin flujos salientes (actividad final).

El concepto de actividad anidada se refiere a una actividad que tiene estructura interna definida por su propio FT. Por tanto, en este modelo se realiza una definición recursiva de las actividades y/o flujos de trabajo asociados.

En los proyectos reales se producen repeticiones de grupos de actividades, es decir, iteraciones. En este modelo de FT's estructurados la estructura de iteración se representa incorporando una condición de iteración al FT (es decir, al grupo de actividades que incluye) que se quiere repetir. El FT definido por dicho sub-grafo se repetirá mientras se cumpla la condición de iteración.

Este modelo de FT's es más sencillo que otros propuestas. Por ejemplo, otros modelos definen explícitamente las condiciones de transición “And-split” y “And-join” que en este modelo se sustituyen por actividades (Sadiq y Orłowska, 1996). Otro ventaja de este modelo es que los FT's son estructurados según la definición de Kiepuszewski et al (2000), que básicamente consiste en que cada or-split tiene su correspondiente or-join y viceversa. Esto tiene la ventaja de que dichos FT's cumplen ciertas propiedades formales útiles con fines de verificación.

5.5.4.2. Estados de un Proyecto y sus Actividades.

Las instancias del concepto de proyecto (definido en la ontología del mantenimiento) se refieren a casos de proyectos del mundo real sujetos a restricciones concretas de tiempo, costes y recursos. Como se comentará en el capítulo 7 en detalle, en el Entorno MANTIS se propone utilizar un SGFT como motor de procesos para dar soporte a la reificación de dichas instancias de proyecto. Por esta razón, el modelo de estados para las instancias de proyectos usado en MANTIS es una adaptación del catálogo de estados definido para las instancias de procesos por la WfMC (1999).

Se han definido tres niveles de detalle para la especificación del estado de una instancia de proyecto, con los siguientes valores correctos:

- *Abierto* (open): la instancia del proyecto está siendo reificada.
 - *Abierto.En-ejecución* (running): la instancia está siendo ejecutada, es decir, alguna instancia de actividad del proyecto está siendo o ha sido ejecutada.
 - *Abierto.No-en-ejecución* (notRunning): temporalmente, la instancia no se está ejecutando.
 - *Abierto.No-en-ejecución.No-comenzado* (notStarted): la instancia ha sido creada en el Entorno pero no ha comenzado todavía su ejecución (la de alguna actividad).
 - *Abierto.No-en-ejecución.Suspendido* (suspended): la ejecución de la instancia del proyecto ha sido suspendida temporalmente.
- *Cerrado* (closed): La reificación de la instancia del proyecto ha finalizado.
 - *Cerrado.Abortado* (aborted): La reificación de la instancia del proyecto ha sido abortada, es decir, el proyecto se ha acabado antes de su conclusión exitosa.
 - *Cerrado.Terminado* (terminated): La reificación de la instancia del proyecto ha concluido con normalidad, es decir, el proyecto se ha terminado con éxito.

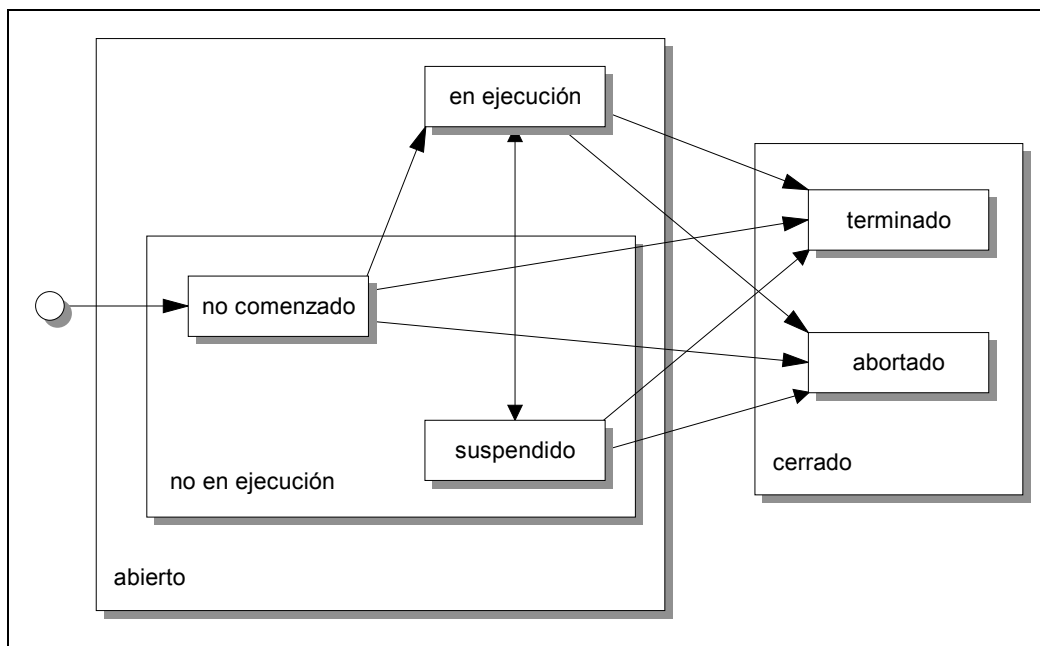


Figura 5-19. Modelo de estados de las instancias de proyecto.

En el diagrama de la Figura 5-19 se muestran estos estados y las transiciones de estado posibles. Aunque sólo se indican las transiciones al nivel más bajo, las correspondientes a niveles superiores también son posibles, por ejemplo, entre “Abierto.No-en-ejecución” y “Abierto.En-ejecución” o entre “Abierto” y “Cerrado”.

Cuando una instancia de un proyecto ha pasado al estado “Abierto”, las instancias de actividades pertenecientes a dicho proyecto pueden tener uno de los siguientes estados básicos (ver Figura 5-20):

- *Inactivo*: la instancia de actividad ha sido creada, dentro de la instancia del proyecto, pero todavía no ha sido activada, es decir, no se está ejecutando.
- *Activo*: la actividad está ejecutándose.
- *Suspendido*: la instancia de actividad está “latente”, su ejecución está parada temporalmente.

Completado: la ejecución de la instancia de actividad ha sido completada.

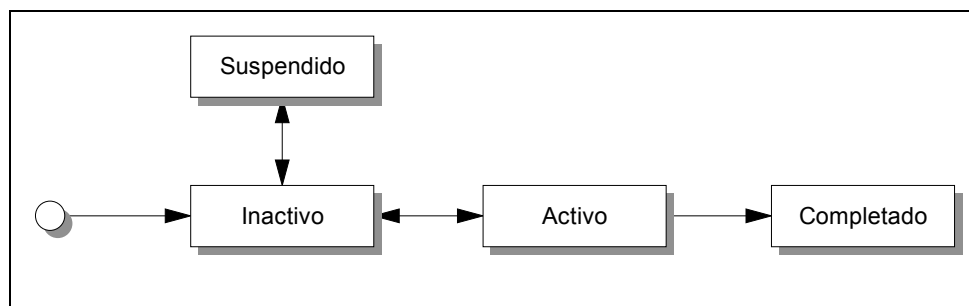


Figura 5-20. Estados y transiciones para instancias de actividad.

5.5.4.3. Representación.

Para la WfMC un FT es “la automatización de un proceso de negocio completo o en parte”. Trasladando esta definición al Entorno MANTIS, es decir, a la automatización de la gestión de proyectos de mantenimiento del software, en la ontología de los flujos de trabajo se han tenido en cuenta las siguientes consideraciones:

- En función de su estructura interna, existen dos tipos de actividades: simples o atómicas (aquellas que no tienen estructura interna a efectos de gestión del proyecto) y complejas o anidadas, cuya estructura interna está definida en base al modelo de FT’s de Sadiq y Orłowska (1999) presentado en el apartado 5.5.4.1. La ejecución de una tarea compleja supone la ejecución del FT subyacente en la misma.
- Una actividad compleja contiene una colección de “nodos” y “flujos de control” interrelacionados.
- Los nodos pueden ser actividades (sub-actividades de la actividad compleja) o condiciones de transición. De esta manera, se está utilizando una definición recursiva de la estructura de una actividad: una actividad principal (por ejemplo, cada una de las definidas en la metodología MANTEMA o en el modelo del PMS de ISO) tiene una estructura interna modelada por un FT que incluye sub-actividades como nodos, que a su vez pueden ser

complejas y ser modeladas por otros FT's que a su vez incluyen otras sub-actividades, y así sucesivamente hasta llegar al nivel de descomposición de trabajo que interese para la gestión del proyecto, es decir, hasta obtener sub-actividades simples. Las condiciones de transición pueden ser una “bifurcación” o una “fusión” (ver Figura 5-18).

- Un flujo de control representa una relación de precedencia temporal entre dos nodos, identificados por las relaciones “empieza-en” y “acaba-en”. Un flujo de control puede ser de diversos tipos según la clase de relación de precedencia que representa: “acabar para empezar”, “empezar para acabar”, “empezar para empezar”, “acabar para acabar”, etc.¹⁹
- Las estructuras iterativas (bucles que repiten la ejecución de una actividad más de una vez) se representan, según el modelo explicado en el apartado 5.5.4.1, mediante los atributos iteración y condición de iteración de las actividades. La existencia de actividades con iteración implica la posibilidad de que durante la reificación de una instancia de un proyecto podrán existir varias reificaciones diferentes de una misma instancia de actividad.
- El estado de ejecución de un proyecto y sus correspondientes actividades se representa mediante atributos de estado en ambos conceptos que toman los valores definidos en los respectivos modelos de estados (ver apartado 5.5.4.2).

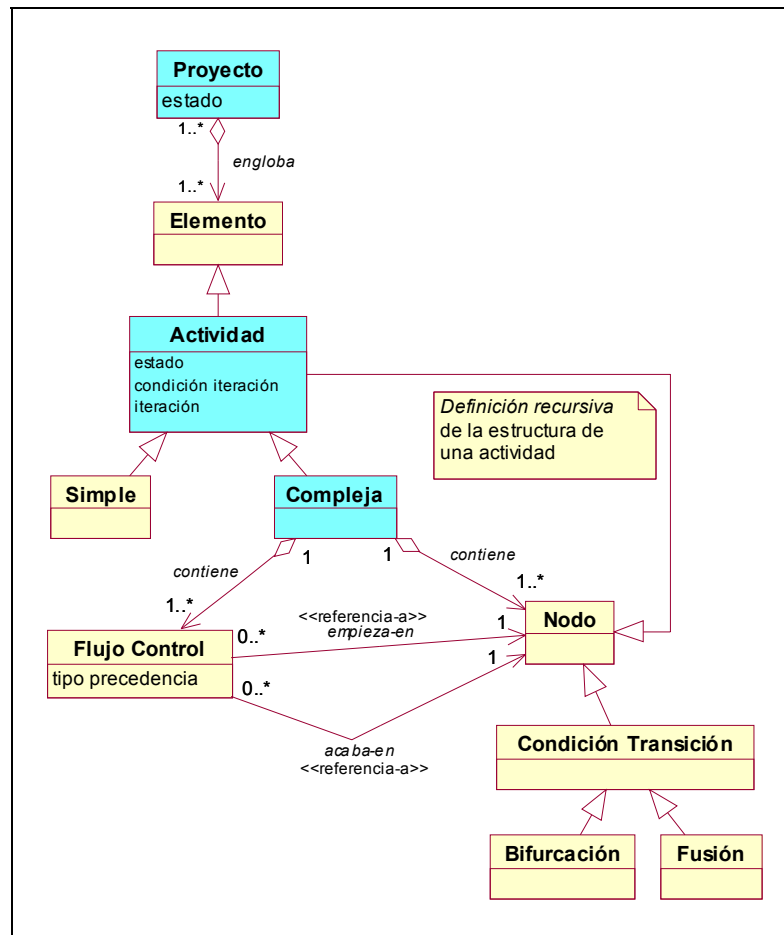


Figura 5-21. Diagrama de la ontología de los flujos de trabajo.

¹⁹ Aunque estos son los tipos de relaciones de precedencia definidos por PMI (2000) y disponibles en la mayoría de herramientas de gestión de proyectos y SGFT's, en realidad es posible definir cualquier otro tipo mediante otras restricciones diferentes.

En la Figura 5-21 se representan todas estas consideraciones mediante los diagramas UML ya utilizados anteriormente. A continuación se presentan el glosario de conceptos y las tablas de atributos e interrelaciones. La tabla de clases de interrelaciones no se incluye porque sólo se considera la clase “referencia-a”, que ya se definió en la ontología del mantenimiento (ver Tabla 5-18).

Concepto	Super-Concepto	Descripción	Propósito
Actividad	Elemento + Nodo	Ver sub-ontología de las actividades.	-
Bifurcación	Condición Transición	Una condición de transición que divide un camino de ejecución de un FT en varios caminos alternativos, entre los cuales se debe seguir uno solo. Sinónimo: Or-split.	Definir estructura de las actividades.
Compleja	Actividad	Actividad que tiene estructura interna (representada por un FT) utilizada durante la gestión del proyecto.	Definir estructura de las actividades.
Condición Transición	Nodo	Un nodo que no es una actividad.	Definir estructura de las actividades.
Elemento	-	Ver ontología del mantenimiento.	-
Flujo Control	Concept	Una relación de precedencia entre dos nodos del FT asociado a una actividad compleja.	Definir estructura de las actividades.
Fusión	Condición Transición	Una condición de transición que unifica varios caminos de ejecución alternativos de un FT en uno solo. Sinónimo: Or-join.	Definir estructura de las actividades.
Nodo	Concept	Una sub-actividad o una condición de transición que aparece como nodo en el FT que representa la estructura interna de una actividad compleja.	Definir estructura de las actividades.
Proyecto	-	Ver ontología del mantenimiento.	-
Simple	Actividad	Actividad que no tiene estructura interna a efectos de gestión del proyecto. Sinónimo: Atómica.	Definir estructura de las actividades.

Tabla 5-19. Ontología de los flujos de trabajo: glosario de conceptos.

Concepto	Atributo	Descripción	Cardinalidad
Actividad	Condición Iteración	Las sub-actividades que forman una actividad iterativa se estarán ejecutando de forma repetida mientras la condición se cumpla.	1
	Estado	Estado de una instancia de una actividad en cuanto a su reificación. Ejemplos: “activo”, “suspendido”, etc. (ver apartado 5.5.4.2).	1
	Iteración	Valor booleano que indica si una actividad es una estructura iterativa y, por tanto, se podrán tener varias reificaciones para una misma instancia de actividad.	1
Flujo Control	Tipo Precedencia	Una expresión condicional que representa el tipo de relación de precedencia entre dos nodos. Ejemplos: “acabar-para-empezar”, “acabar-para-acabar”.	1

Concepto	Atributo	Descripción	Cardinalidad
Proyecto	Estado	Estado de una instancia de un proyecto en cuanto a su reificación. Ejemplos: “abierto”, “cerrado”, “abierto.en-ejecución”, etc. (ver apartado 5.5.4.2).	1

Tabla 5-20. Ontología de los flujos de trabajo: tabla de atributos.

Nombre	Descripción
Engloba	Ver ontología del mantenimiento.
Contiene	Una actividad compleja tiene una estructura interna definida por un FT, es decir, contiene una colección de nodos y flujos de control interrelacionados.
Empieza-en	Un flujo de control empieza en un nodo (origen)
Acaba-en	Un flujo de control acaba en un nodo (destino).

Tabla 5-21. Ontología de los flujos de trabajo: tabla de interrelaciones.

5.5.5. Ontología de la Medida.

Medición es el proceso por el cual números o símbolos son asignados a atributos de entidades de forma que los describen de manera claramente definida (Filkenstein, 1984).

El propósito del proceso de medición del software es “coleccionar, analizar y proveer los datos ... que son necesarios para una gestión efectiva de los procesos y, objetivamente, demostrar la calidad de los productos” (ISO/IEC, 1998c) y, puesto que el Entorno MANTIS pretende ayudar a la gestión integrada del PMS, parece obvio el interés de incluir en MANTIS el proceso de medición. Satisfacer este interés pasa por integrar dicho proceso en el marco de trabajo conceptual. Con esta finalidad se ha diseñado la ontología de la medida que se presenta a continuación. La importancia asignada al proceso de medición en MANTIS está en consonancia con la idea elemental de que “*gestionar es medir*”²⁰.

La integración en el Entorno MANTIS del proceso de medición como uno de los aspectos fundamentales para la gestión de los proyectos de mantenimiento hace más factible una adecuada implementación de dicho proceso y, en consecuencia, más fácil conseguir que:

- se establezca y mantenga un compromiso organizacional para la medición;
- se identifiquen las necesidades de información de los procesos técnicos y de gestión;
- se definan y desarrollen un conjunto adecuado de métricas basadas en las necesidades de información;
- las actividades de medición sean identificadas;
- los datos requeridos sean coleccionados, almacenados y analizados, y los resultados interpretados;
- la información obtenida pueda ser utilizada para la toma de decisiones y provea una base objetiva para la comunicación;
- el proceso de medición y las métricas sean evaluados; y

²⁰ El lector puede comprobar la extensión de esta idea consultando qué entienden los usuarios del famoso buscador *Google* por gestionar: http://www.googlism.com/who_is/m/managing/.

- las propuestas de mejoras sean comunicadas al responsable del proceso.

Adicionalmente a lo referido a continuación, en el capítulo 6 de esta tesis se ha incluido la utilización de esta ontología, junto con el resto del marco de trabajo del Entorno MANTIS, para la evaluación y mejora del proceso de mantenimiento del software.

La ontología de la medida de MANTIS ha sido elaborada a partir de las tres fuentes documentales siguientes (códigos FD7a, FD7b y FD4 en la Tabla 5-3):

- El proceso de medición del software del borrador de la propuesta de estándar ISO 15939 (ISO/IEC, 2002).
- El modelo conceptual para representar colecciones de datos software de Kitchenham et al (2001a).
- El modelo conceptual de medición del software propuesto por Becker-Kornstaedt y Webby (1999).

Entre estas fuentes, el futuro estándar ISO 15939 ha sido la más utilizada. Las principales aportaciones de dicho estándar a la ontología han sido las siguientes:

- a) la jerarquía de actividades y tareas que conforman el proceso de medición del software;
- b) el “*Measurement Information Model*”, que se ha utilizado y adaptado como base para el modelo conceptual comentado después; y
- c) la nomenclatura propuesta (aunque ha sido adaptada ligeramente para poder realizar una integración adecuada con las demás ontologías de MANTIS).

5.5.5.1. El Proceso de Medición.

Dentro del sistema de procesos de MANTIS, el proceso de medición contemplado en esta ontología pertenece al subsistema organizacional, categoría de organización (ver apartado 5.4). La importancia de definir adecuadamente este proceso de medición, y los datos y métricas correspondientes, ya ha sido manifestada por diversos autores. Por ejemplo, Olsina et al (2002) han presentado recientemente un marco conceptual para la definición y explotación de métricas de calidad, que incluye un modelo conceptual que pretende servir para representar cualquier tipo de métrica. También está en esta línea la ontología de la medición de Kim (1999) ya que, aunque se enmarca dentro de un modelo de calidad para procesos tradicionales de fabricación industrial basado en ISO 9000, algunas de sus aportaciones son trasladables al Entorno MANTIS, y a la medición en ingeniería del software en general.

Las actividades y tareas que forman el proceso de medición propuesto en MANTIS, basadas en la propuesta de estándar de ISO/IEC (2002), son las siguientes:

Proceso de Medición

- Establecer y mantener el compromiso de medición.
 - o Aceptación de los requerimientos de medición.
 - o Asignar recursos.
- Planificar el proceso.

- o Caracterizar la unidad organizacional.
- o Identificar las necesidades de información.
- o Seleccionar métricas.
- o Definir los procedimientos de recolección, análisis y distribución de datos.
- o Definir criterios para evaluar los resultados y el proceso de medición.
- o Revisar, aprobar y facilitar recursos para las tareas de medición.
- o Adquirir e implantar las tecnologías de soporte.
- Realizar las mediciones.
 - o Integrar procedimientos.
 - o Recoger datos.
 - o Analizar los datos y elaborar los productos informativos.
 - o Comunicar resultados.
- Evaluación.
 - o Evaluar los productos informativos y el proceso de medición.
 - o Identificar mejoras potenciales.

Las cuatro actividades referidas forman un ciclo iterativo posibilitando la realimentación y mejoras continuas. Dentro de cada actividad, las tareas también son iterativas. En la Figura 5-22 se ha representado brevemente este modelo cíclico. Los conceptos incluidos en dicha figura (productos informativos, necesidades de información, etc.) se definen en los apartados próximos. En realidad, este modelo del proceso de medición es una adaptación del ciclo “*Plan-Do-Check-Act*” propuesto en la norma 9001 de ISO (2000).

El proceso está dirigido por las necesidades de información de la organización. Para cada necesidad de información, el proceso produce un producto informativo que satisface dicha necesidad. El producto informativo es la base para la toma de decisiones por la organización.

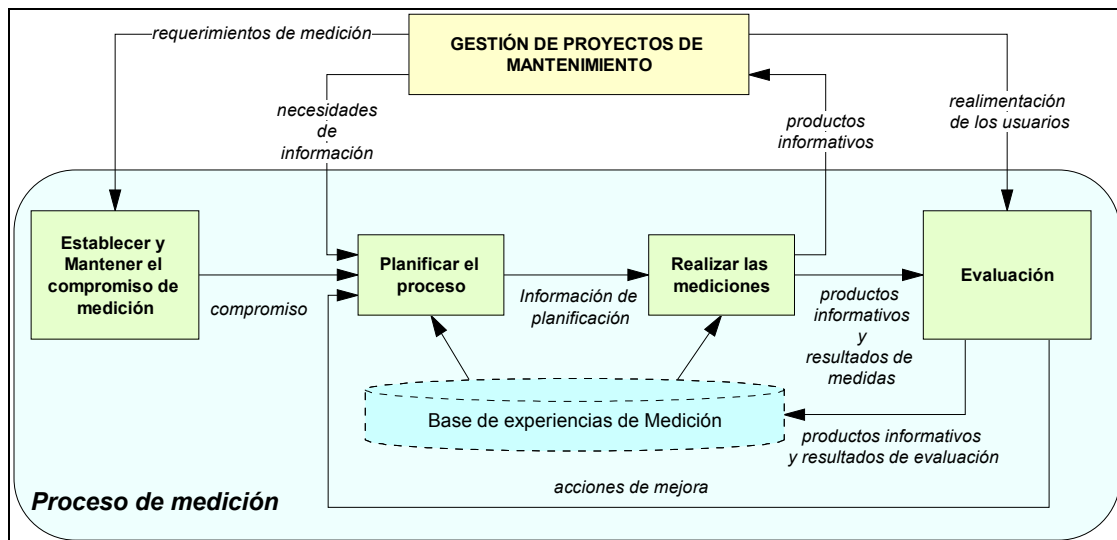


Figura 5-22. Modelo del Proceso de Medición propuesto en MANTIS.

5.5.5.2. Descripción Conceptual de la Medición.

En este apartado realizamos una descripción de que se entiende por **medir** en la ontología de la medida de MANTIS, es decir, las estructuras y relaciones existentes entre las necesidades de información de un proyecto de mantenimiento y los elementos relevantes (artefactos, actividades, recursos y agentes) en la gestión de dicho proyecto, los cuales permiten obtener los productos informativos que satisfacen dichas necesidades de información. Las relaciones que caracterizan este modelo son las siguientes:

- La selección o definición de las métricas adecuadas para satisfacer una necesidad de información comienza con un concepto medible, es decir, una idea de cuales son los atributos medibles relacionados con la necesidad de información y cómo están relacionados entre sí.
- El valor de un atributo de un elemento puede ser medido por una o varias métricas (también llamadas medidas). Cada observación o medición²¹ permite obtener el valor de un atributo en un determinado instante o punto en el tiempo para una de sus métricas asociadas. Cada métrica tiene asociada una determinada unidad de medición que pertenece a una determinada escala. Las escalas pueden ser de 4 tipos diferentes: nominal (valores categóricos no ordenados), ordinal (valores categóricos ordenados), intervalo (igual distancia en los valores medidos corresponde a igual cantidad del atributo), y ratio (igual que intervalo pero con el valor 0 indicando que no hay ninguna cantidad del atributo).
- Las observaciones pueden ser directas o indirectas. Las directas consisten en utilizar alguna de las métricas base asociadas con un atributo. Cada métrica base está definida por un determinado método de medición (operaciones que establecen una correspondencia entre el atributo y la escala asociada a la métrica). Los métodos pueden ser subjetivos u objetivos.
- Para realizar observaciones indirectas se dispone de métricas derivadas y de indicadores. Se diferencian entre sí en que las primeras no son directamente utilizables para satisfacer las necesidades de información mientras que los segundos sí lo son. El valor de una métrica derivada se obtiene aplicando una función de medición a dos o más valores de métricas base.
- Los indicadores pueden tener diferentes niveles o grados de exactitud. Por esta razón, el valor de un indicador puede ser cuantitativo (numérico) o cualitativo (categórico). El valor de un indicador se obtiene aplicando un cierto modelo de análisis, es decir, un algoritmo para combinar los valores de una o varias métricas definidas (métricas derivadas y/o base) utilizando unos determinados criterios de decisión.
- Una interpretación consiste en una explicación, en forma comprensible para los interesados, del resultado de una observación de un indicador.
- Un producto informativo es el resultado del proceso de medición que satisface una necesidad de información. Está formado por la colección de interpretaciones adecuadas.

5.5.5.2.1. Ejemplo: Actualizar el Importe de un Servicio de Mantenimiento.

Un ejemplo concreto de aplicación de esta descripción conceptual es el siguiente: Una empresa (mantenedor) lleva a cabo proyectos de mantenimiento de software para terceros. El mantenedor realiza dichos proyectos sujetos a unos contratos de un año de duración que establecen unos acuerdos de nivel de servicio y unos ingresos económicos a cambio. Cuando se acerca la fecha de fin de un contrato, el mantenedor tiene que decidir si los ingresos percibidos están en consonancia con los recursos dedicados a mantener el producto correspondiente o, por el contrario, deben incrementarse o reducirse. El concepto medible es que el esfuerzo de mantenimiento está en relación con el tamaño del software mantenido y con la cantidad de código fuente modificado. Utilizando diversas métricas base y métricas derivadas se obtiene un indicador de la cantidad de mantenimiento necesaria para el producto X. Este indicador es utilizado por los gestores para preparar un producto informativo indicando los costes del mantenimiento durante el último año.

²¹ Hemos elegido preferentemente la nomenclatura “observación” en vez de medición para evitar el equívoco que se produce con medición como proceso (concepto de clase actividad).

Para este supuesto, los conceptos presentados anteriormente se podrían concretar en lo siguiente:

- *Necesidad de información*: un directivo formula una solicitud con la pregunta ¿es adecuado el importe actual del servicio de mantenimiento del producto X?.
- *Concepto medible*: “Esfuerzo de mantenimiento”.
- *Atributos y elementos relevantes*: “Tamaño” (referido a los elementos de tipo “Componente software”); y “Complejidad” (referido a los elementos de tipo “Petición de mantenimiento”).
- *Observaciones*: Se debe realizar una observación directa del atributo “Tamaño” usando la métrica “Líneas de código fuente” para cada componente del producto X. También se debe realizar una observación directa del atributo “Complejidad” usando la métrica “Número de líneas de código fuente modificadas” para cada petición de mantenimiento del producto X. Además, se realizarán tres observaciones indirectas, dos para las dos métricas derivadas y una para el indicador, que se detallan después.
- *Métricas base*: “Líneas de código fuente del componente C” (para el tamaño de cada componente); y “Número de líneas de código fuente modificadas para atender la petición de mantenimiento P” (para la complejidad de cada petición de mantenimiento).
- *Métodos de medición*: Contar el número de líneas de código fuente utilizando la herramienta SoftMetric (para el tamaño); y contar el número de líneas de código fuente modificadas (es decir, cambiadas, añadidas o eliminadas) por una petición de mantenimiento utilizando la herramienta de gestión de configuración. Ambos métodos de medición son de tipo objetivo.
- *Escalas*: Ambos métodos de medición están asociados con la escala de números enteros desde 0 a infinito, que es de tipo ratio.
- *Unidades de medición*: LDCF (línea de código fuente) para la primera métrica base; y LDCFM (línea de código fuente modificada) para la segunda.
- *Métricas derivadas*: “Tamaño del producto X”; y “Cantidad de código fuente modificado del producto X”.
- *Funciones de medición*: $\text{Tamaño}(\text{producto X}) = \text{suma}(\text{LDCF de componentes C de X})$; y $\text{Cantidad-código-modificado(X)} = \text{suma}(\text{LDCFM de peticiones P de X})$.
- *Indicador*: “Cantidad de mantenimiento”, que indica el número de horas-persona dedicados a mantener un producto software. Este indicador es de tipo numérico entero.
- *Modelo*: por medio de una base de experiencias de proyectos anteriores, la organización ha elaborado una tabla que permite obtener la cantidad de mantenimiento del producto X, en horas-persona, a partir de los valores de tres datos: “tipo de producto de X” (de gestión en entorno PC, de gestión en entorno Host, cliente-servidor web, etc.), “tamaño de X” y “cantidad de código modificado de X”.
- *Criterios de decisión*: Son una serie de reglas que permiten clasificar un producto X en alguno de los tipos de producto de la tabla. En su defecto se podrá utilizar una media ponderada de los valores de más de un tipo.
- *Interpretación*: La interpretación incluirá el coste medio por hora-persona actual de la organización, y el coste económico en Euros de la cantidad de mantenimiento obtenida, que será la multiplicación de los valores anteriores.
- *Producto Informativo*: Los gestores económicos elaboran un informe explicando en lenguaje natural los costes económicos y en horas-persona del mantenimiento realizado al producto X el último año y los ingresos obtenidos a cambio. Con dicho informe los

directivos podrán tomar la decisión de subir, bajar o dejar igual el importe de dicho mantenimiento.

5.5.5.3. Representación.

La ontología de la medida (Género et al, 2003), basada en el modelo del proceso de medición y la descripción conceptual comentados, debe integrarse con las ontologías del mantenimiento y de los flujos de trabajo. Para ello se han realizado las siguientes consideraciones:

- La gestión de un proyecto de mantenimiento de software tiene unas determinadas necesidades de información. Utilizar la idea de proyecto como uno de los conceptos base de esta ontología permite su integración con las dos ontologías anteriores. Esta opción también es la postulada por Kitchenham et al (2001a) para representar datos de mediciones del software.
- El proceso de medición recibe como entradas necesidades de información para producir como salidas productos informativos. El proceso de medición es una especialización del concepto genérico de actividad (ver sub-ontología de las actividades). En la jerarquía de actividades de la Figura 5-12 se incluiría como especialización de Actividad-Principal-Gestión.
- Un producto informativo es un tipo especial de artefacto (ver sub-ontología de los productos). Para elaborar el producto informativo que satisface una necesidad de información se sigue el camino descrito en el modelo conceptual: a cada necesidad de información le corresponde un concepto medible que está asociado con uno o varios atributos medibles; cada atributo es calificado con una o varias observaciones, cada una de las cuales se refiere a una determinada métrica; de dichas observaciones y métricas se obtienen una o varias interpretaciones, que forman el producto informativo. Este camino se ha destacado en la Figura 5-23 poniendo en color azul las interrelaciones (asociaciones, agregaciones o generalizaciones) y los conceptos que lo representan.
- Los atributos medibles describen propiedades o características de un elemento (artefacto, actividad, recurso o agente) que es significativo para los fines de gestión de un proyecto.
- Cada atributo puede ser medido con una o varias métricas. Por ejemplo, el tamaño de un componente software puede ser medido con las métricas “líneas de código fuente” o “número de puntos función”. Los atributos medibles son calificados mediante observaciones que permiten obtener su valor, para alguna de las métricas asociadas, en un determinado instante con una cierta precisión. Estas observaciones coinciden con el concepto de punto de medición en la ontología de gestión de la calidad de Kim (1999).
- Cada métrica tiene un valor por defecto que es el único conocido antes de realizar alguna observación basada en dicha medida.
- Los métodos de medición, funciones de medición y modelos de análisis son procedimientos de la clase Método (ver sub-ontología de organización del proceso – procedimientos); aunque en algunos casos también se podrían incluir en la clase de Técnicas.
- Mientras que las medidas base suponen mediciones directas del valor de un atributo, las medidas derivadas y los indicadores suponen mediciones indirectas. Por otro lado, mientras que las medidas base o derivadas son medidas definidas, es decir, existe un método para obtener su valor exacto, los indicadores pueden ser medidas sujetas a inexactitudes. Pueden ocurrir situaciones concretas en que una medida derivada, o incluso una medida base, sirva

a la vez de indicador, es decir, se le pueda asociar una interpretación para darle utilidad como parte de un producto informativo.

- Para obtener un producto informativo es necesaria, al menos, una observación directa para disponer del valor de una métrica base. También es necesaria una observación indirecta para obtener el valor de un indicador que permita elaborar, al menos, una interpretación.
- Las unidades y escalas juegan un papel equivalente al de los tipos de atributos del modelo conceptual para la medición de Becker-Kornstaedt y Webby (1999), pero con la ventaja de que la opción elegida permite que las mediciones sean repetibles, ya que se especifican el elemento del proyecto que se quiere medir, el atributo que se quiere medir, la unidad de medición (y escala asociada), y el procedimiento de medición (Kitchenham et al, 1995)²².

Esta ontología define conceptos a un nivel lo más genérico posible con el objetivo de que sea utilizable para distintos modelos y métodos de medición. Así, el modelo de representación de datos software de Kitchenham et al (2001a) o el modelo conceptual para el dominio de métricas propuesto por Olsina et al (2002), son utilizables con ella. El único problema (de nomenclatura) surge en este último caso; ya que Olsina et al distinguen entre métrica (lo que nosotros hemos llamado métrica o medida de forma sinónima) y medida, que se corresponde con el concepto de observación en nuestra ontología. Con el fin de clarificar esta confusión, en la Tabla 5-22 se muestra el resumen de la nomenclatura utilizada por diversos autores para los conceptos de métrica-medida y observación. Además, existe el problema de que en inglés se utiliza la misma palabra (*measure*) para dos significados diferentes: acción (verbo) y resultado de la acción (nombre). Para evitar los riesgos de este tipo o similares, se ha optado por dejar los términos en el idioma original de los trabajos consultados.

Fuente	Métrica	Observación
MANTIS	Métrica o Medida	Observación o Medición
(ISO/IEC, 2002)	Measure	Observation (sólo para métricas base)
(Kitchenham et al, 2001a)	Element Measure Type	Recorded Value
(Becker-Kornstaedt y Webby, 1999)	no incluido	Value
(Olsina et al, 2002)	Métrica	Medida
(Kim, 1999)	Measure	Measurement Point

Tabla 5-22. Diferentes nomenclaturas para los conceptos de métrica y observación.

En la Figura 5-23 se muestra el diagrama UML de la ontología de la medida. Los conceptos propios de esta ontología se muestran en color amarillo o azul mientras que los conceptos “heredados” de las dos ontologías anteriores se muestran en color blanco.

²² Aunque, tal como proponen estos autores, en esta ontología no se incluye quién (el rol) es el responsable de las actividades de medición, es porque dicha información ya se ha contemplado de forma global en la sub-ontología de los agentes anteriormente presentada.

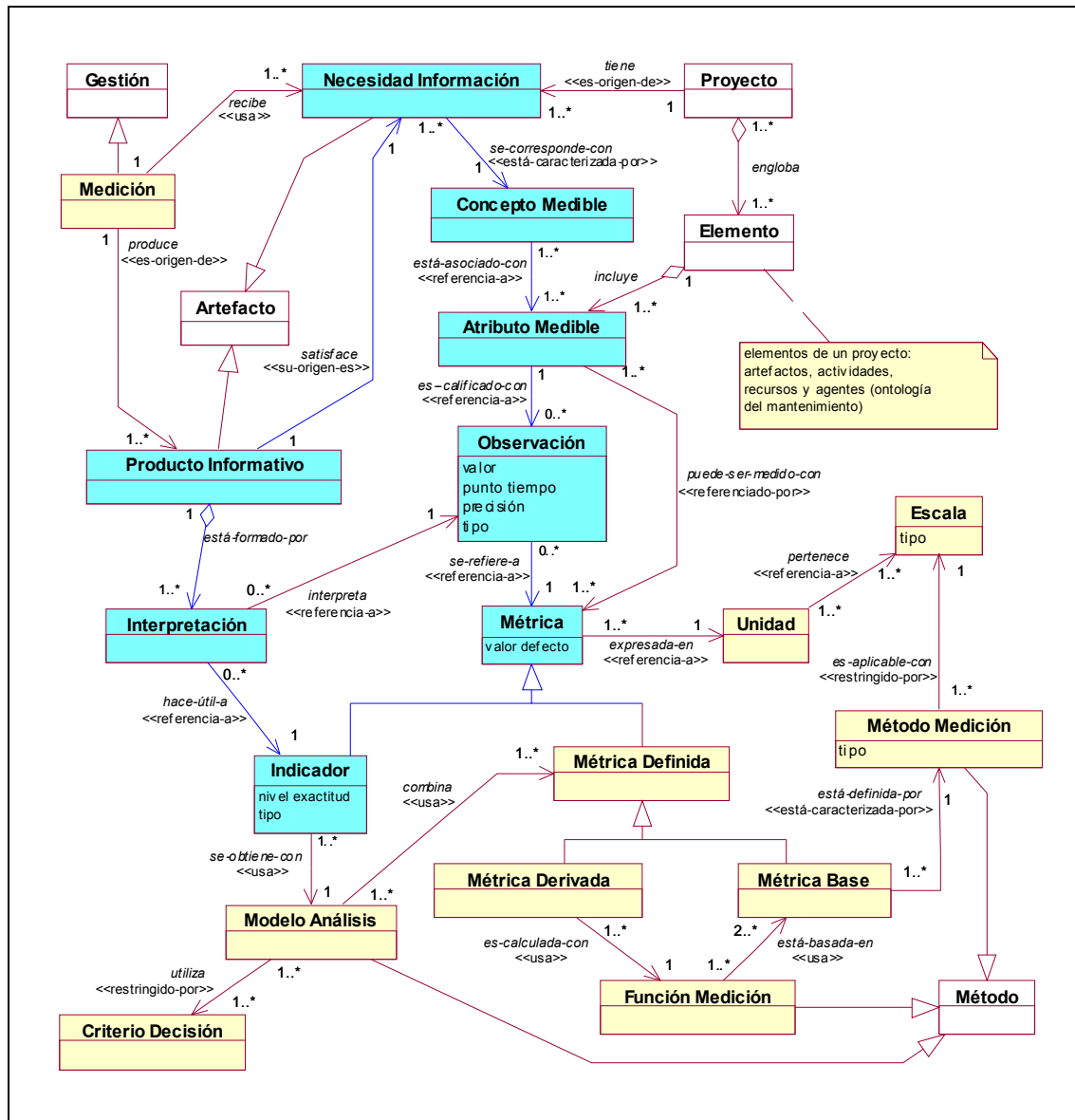


Figura 5-23. Diagrama de la ontología de la medida.

A continuación se incluyen las diversas tablas en la versión adaptada de REFSENSO ya utilizada en las anteriores ontologías de MANTIS. Todas las interrelaciones son de clases ya incluidas en la Tabla 5-18 de clases de interrelaciones, por tanto, no es necesario añadir dicha tabla. Esto se puede comprobar mirando los estereotipos asociados a dichas interrelaciones (en el diagrama de la Figura 5-23 se muestran como asociaciones estereotipadas).

Concepto	Super-Concepto	Descripción	Propósito
Artefacto	-	Ver sub-ontología de los productos.	-
Atributo Medible	Concept	Propiedad o característica de un elemento que puede ser distinguida cualitativa o cuantitativamente.	Identificar objetos a medir.

Concepto	Super-Concepto	Descripción	Propósito
Concepto Medible	Concept	Idea que establece una relación abstracta entre atributos y necesidades de información. Ejemplos: calidad, rendimiento, madurez, capacidad.	Identificar objetos a medir.
Criterio Decisión	Concept	Valores umbral, objetivos o patrones usados para determinar la necesidad de una acción o para describir el nivel de bondad del valor de una métrica. Es una ayuda para interpretar los resultados de las mediciones.	Medir.
Elemento	-	Ver ontología del mantenimiento. Elemento de un proyecto (artefactos, actividades, recursos o agentes) que será caracterizado midiendo alguno de sus atributos. Sinónimo: Entidad.	-
Escala	Concept	Conjunto ordenado de valores, continuo o discreto, o un conjunto de categorías a los que está asociada una métrica de un atributo.	Medir.
Función Medición	Método	Algoritmo o cálculo realizado para combinar dos o más valores de métricas base para obtener el valor de una métrica derivada. La unidad y escala de una métrica derivada depende de cual sea está función de medición.	Establecer modo de medir.
Gestión.	-	Ver sub-ontología de las actividades.	-
Interpretación.	Artefacto	Artefacto de tipo documento con una explicación del resultado de una observación de un indicador para satisfacer una necesidad de información.	Gestionar un proyecto.
Indicador	Métrica	Métrica que provee una estimación (evaluación) de unos determinados atributos con respecto a ciertas necesidades de información. Un indicador es la base para la toma de decisiones y análisis de un proyecto. El valor de un indicador puede ser numérico o categórico (cualitativo).	Medir.
Medición	Gestión	Proceso de Gestión consistente en realizar las actividades necesarias para obtener los productos informativos que satisfagan las necesidades informativas originadas durante la gestión de un proyecto.	Gestionar un proyecto.
Métrica	Concept	Variable asociada a uno o varios atributos que permite cualificarlos mediante observaciones (asignación de valores). Puede ser una métrica base, una métrica derivada o un indicador. Sinónimo: Medida.	Medir.
Métrica Base	Métrica Definida	Métrica definida en términos de un atributo y el método de medición para obtener su valor. Sinónimo: Métrica directa.	Medir.
Métrica Definida	Métrica	Métrica para la que existe una forma matemática exacta de obtener su valor. Puede ser una métrica base o una métrica derivada.	Medir.
Métrica Derivada	Métrica Definida	Métrica definida en función de 2 o más valores de métricas base. Capturan información sobre más de un atributo o del mismo para más de un elemento.	Medir.
Método	-	Ver sub-ontología de organización del proceso.	-
Método Medición	Método	Método consistente en una secuencia lógica de operaciones usadas para cuantificar un atributo respecto a una cierta escala. Existen dos tipos de métodos de medición: subjetivos (basados en el juicio humano) y objetivos (basados en reglas numéricas). Ejemplos: contar ocurrencias u observar el paso del tiempo.	Establecer modo de medir.
Modelo Análisis	Método	Algoritmo o cálculo que combina una o más métricas definidas con ciertos criterios de decisión para obtener el valor de un indicador y su interpretación.	Establecer modo de medir.

Concepto	Super-Concepto	Descripción	Propósito
Necesidad Información	Artefacto	Artefacto de tipo documento describiendo información necesaria para gestionar un proyecto (sus objetivos, hitos, riesgos y problemas).	Gestionar un proyecto.
Producto Informativo	Artefacto	Artefacto de tipo documento que permite satisfacer una cierta necesidad de información. Para ello, incluye las interpretaciones de las observaciones de uno o varios indicadores.	Gestionar un proyecto.
Observación	Concept	Acción que ocurre en un determinado instante o punto en el tiempo que permite obtener el valor de un atributo referido a una métrica. Ejemplo: El tamaño del módulo CST el día 1-1-2002 es de 12500 LDCF.	Medir.
Proyecto	-	Ver ontología del mantenimiento.	-
Unidad	Concept	Cantidad, definida o adoptada por convención, utilizada para comparar cantidades de la misma clase. Sólo cantidades expresadas en la misma unidad son directamente comparables. Ejemplos: líneas de código, horas, personas-mes.	Medir.

Tabla 5-23. Ontología de la medida: glosario de conceptos.

Concepto	Atributo	Descripción	Cardinalidad
Escala	Tipo	Indica la naturaleza de la relación entre los valores de la escala. Puede ser de 4 tipos: nominal, ordinal, intervalo y ratio.	1
Indicador	Nivel Exactitud	Una cuantificación de la incertidumbre o exactitud inherente a la manera de obtener el valor de un indicador.	1
	Tipo	Naturaleza del valor obtenido de un indicador. Puede ser numérico (cuantitativo) o categórico (cualitativo).	1
Métrica	Valor Defecto	Valor inicial de una métrica antes de que se realice una observación de dicha métrica.	1
Método Medición	Tipo	Indica la naturaleza de las operaciones usadas para cuantificar un atributo. Hay dos tipos: subjetivos (cuantificación influida por juicios humanos) y objetivos (cuantificación basada en reglas numéricas).	1
Observación	Precisión	Indicador de la precisión que tiene el valor de una observación.	1
	Punto Tiempo	Momento o instante en el tiempo en que se ha realizado la observación.	1
	Tipo	Indicador de si la observación es directa (asociada a una métrica base) o indirecta (asociada a una métrica derivada o un indicador).	1
	Valor	Resultado numérico o categórico asignado a una métrica (base, derivada o indicador) de un atributo como resultado de una observación. Sinónimo: Dato.	1

Tabla 5-24. Ontología de la medida: tabla de atributos.

Nombre	Descripción
Combina	Un modelo de análisis combina los valores de una o varias métricas definidas (derivadas o base) para obtener el valor de un indicador.
Engloba	Ver ontología del mantenimiento.
Es-aplicable-con	Un método de medición es aplicable con una determinada escala. Ejemplo: los métodos subjetivos sólo están soportados por escalas ordinales o nominales.
Es-calculada-con	El valor de una métrica derivada se calcula con una determinada función de medición.
Es-calificado-con	Un atributo medible es calificado con una o varias observaciones.

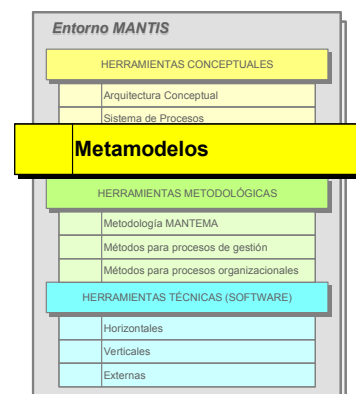
Nombre	Descripción
Está-asociado-con	Un concepto medible está asociado con uno o varios atributos medibles.
Está-basada-en	La función de medición, utilizada para calcular el valor de una métrica derivada, está basada en el valor de 2 o más métricas base.
Está-definida-por	Una métrica base se define en función del método de medición utilizado para obtener su valor.
Está-formado-por	Un producto informativo está formado por una o varias interpretaciones.
Expresada-en	Los resultados obtenidos con una métrica están expresados en una determinada unidad de medición.
Incluye	Un elemento relevante de un proyecto incluye uno o varios atributos medibles.
Pertenece	Una unidad de medición pertenece al conjunto de valores de una o varias escalas (todas ellas del mismo tipo).
Produce	El proceso de medición produce productos informativos como resultado de sus actividades.
Puede-ser-medido-con	Un atributo medible puede ser medido con varias métricas.
Recibe	El proceso de medición, dentro de la gestión de un proyecto, utiliza como entrada de sus actividades las necesidades de información de dicho proyecto.
Satisface	Un producto informativo satisface una necesidad de información.
Se-corresponde-con	Una necesidad de información concreta de un proyecto concreto se corresponde con un concepto medible.
Se-obtiene-con	El valor, nivel de exactitud e interpretación de un indicador se obtienen con un determinado modelo de análisis.
Se-refiere-a	Una observación obtiene el valor de un atributo referido a una determinada métrica.
Tiene	Un proyecto tiene necesidades de información que es necesario satisfacer para su correcta gestión.
Utiliza	Un modelo de análisis utiliza uno o varios criterios de decisión.
Interpreta	Una interpretación interpreta (explica) el resultado de una observación.
Hace-útil-a	Una interpretación aporta utilidad en una necesidad concreta a un indicador.

Tabla 5-25. Ontología de la medida: tabla de interrelaciones.

5.6. Metamodelos.

Las ontologías presentadas en los apartados anteriores describen el conocimiento más importante del dominio del problema abordado en el Entorno MANTIS: la gestión de proyectos de mantenimiento del software. Esta conceptualización explícita es implementada para su utilización por las herramientas software de MANTIS mediante tres tipos de información:

- Los datos de realización de proyectos reales de mantenimiento capturados mediante algunas de las herramientas incluidas en MANTIS u otras herramientas externas con las que interacciona (un SGFT, un software de gestión de proyectos, un CASE de pruebas o gestión de configuración, etc);
- Los modelos concretos que representan los diversos procesos realizados durante la gestión de proyectos de mantenimiento, que se han incluido en el sistema de procesos; y



- c) Los metamodelos que sirven de ayuda para integrar modelos similares.

Estos tres tipos de información se corresponden con los niveles M0, M1 y M2 de la arquitectura conceptual del Entorno MANTIS (ver apartado 5.4), es decir, el conocimiento representado en las ontologías se “distribuye” a nivel de implementación en los tres niveles citados. Ejemplos sencillos de esta manera de “implementar las ontologías” son los siguientes:

- El concepto “Actividad” (apartado 5.5.3.2) pertenecerá al metamodelo general de proceso software en el nivel M2 (con una correspondencia con Clase-MOF del nivel M3).
- El concepto “Tarea NP2.3- Ejecutar pruebas unitarias” (ver anexo B) pertenecerá al modelo de PMS de la metodología MANTEMA en el nivel M1 (con una correspondencia “es-instancia-de” con el concepto de “Actividad” del párrafo anterior).
- La información “Juan es el responsable de la tarea ‘NP2.3- Ejecutar pruebas unitarias’ para la petición de mantenimiento N° 436” pertenecerá a los datos del proyecto concreto correspondiente en el nivel M0 (con una correspondencia con el concepto del párrafo anterior, ya que se trata de asignar valor al atributo ‘responsable’ de una instancia de dicho concepto).

Por tanto, para poder aprovechar este conocimiento ontológico por las herramientas software resulta necesario disponer de metamodelos y modelos adecuados (Ruiz et al, 2001a), (Ruiz et al, 2001b). Los modelos se corresponden con definiciones de métodos concretos, es decir, de las herramientas metodológicas de MANTIS que se comentan en el capítulo 6. En consecuencia, mientras que las ontologías en MANTIS se refieren a todo el conocimiento del dominio del problema (niveles M0, M1 y M2) y sirven, principalmente, para la compartición y reutilización del conocimiento por los agentes humanos, los metamodelos sólo se refieren al nivel M2 y sirven, exclusivamente, para fines de implementación de modelos²³.

A continuación presentamos los dos metamodelos que hemos definido en MANTIS para la representación de modelos de proceso software y para la medición. Además, la arquitectura conceptual en 4 capas permite la incorporación de nuevos metamodelos en caso de ser necesarios (ver Figura 5-24).

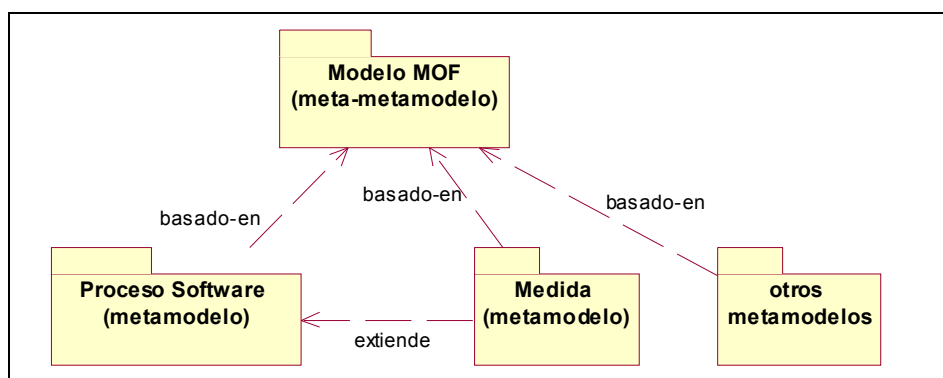


Figura 5-24. Arquitectura general de metamodelado del Entorno MANTIS.

²³ Esta propuesta de uso diferenciado de las ontologías y de los metamodelos en MANTIS ha sido avalada con posterioridad por Spyns et al (2002) que, a juicio de este autor, exponen claramente las diferencias entre el (meta)modelado de datos y la construcción de ontologías.

Estos metamodelos, y cualquier otro que se pueda necesitar, se definen y gestionan en el Entorno MANTIS mediante la herramienta METAMOD (ver capítulo 7).

Se han identificado cerca de 30 usos de metamodelado en el área de los sistemas de información (Castellani, 1998). Entre ellos, los más interesantes para los objetivos de MANTIS son los reactivos a:

- La creación inicial de modelos: representación de varios modelos diferentes; unificación de modelos, y representación de la evolución (de versiones) de modelos.
- La particularización de modelos: extracción de subconjuntos coherentes de modelos.
- Otros usos: diseño de herramientas CASE.

Un trabajo muy interesante que apoya la utilidad e importancia de los metamodelos en los EIS es el desarrollado por Koskinen (1999), que utiliza una aproximación basada en metamodelado para la particularización de modelos de proceso y su posterior reificación. Este autor realiza esta propuesta para su uso en entornos MetaCASE, es decir, aquellos que integran CASE adaptables para soportar métodos evolutivos.

Por razones de legibilidad, los metamodelos se representan mediante diagramas UML, aunque el lenguaje real (abstracto) en que se manejan en el entorno MANTIS es MOF (el anexo I incluye algunos ejemplos), es decir, cada elemento de un metamodelo se define como instancia de alguna clase de constructor del modelo MOF (ver anexo C).

Tal como se ha comentado anteriormente, las ontologías incluidas en este mismo capítulo suponen una ayuda para la definición de estos o cualquier otro metamodelo del dominio del Entorno MANTIS. Por esta razón, la mayoría de las clases y asociaciones que aparecen en los metamodelos se corresponden con conceptos presentados en dichas ontologías. Además, puesto que se trata de metamodelos del nivel M2 de la arquitectura conceptual, los elementos de metamodelado son instancias de los elementos (constructores) del Modelo MOF (nivel M3). Por tanto, las clases de un metamodelo serán instancias de clase-MOF, las interrelaciones serán instancias de asociación-MOF e, igualmente, los paquetes serán instancias de paquete-MOF. Esta correspondencia directa también se da en la representación gráfica mediante diagramas de clase UML: una clase de diagrama UML representa una instancia de clase-MOF; una asociación, agregación o generalización UML también representa una instancia de asociación-MOF; y un paquete UML representa una instancia de paquete MOF. Esta sencillez y fácil correspondencia entre los elementos de las ontologías, los elementos de los metamodelos y los elementos de los diagramas UML -que representan gráficamente dichos metamodelos- es posible gracias al uso combinado de la arquitectura basada en MOF, de las ontologías definidas y de UML como sistema de representación de los metamodelos.

5.6.1. Metamodelo Genérico de Proceso Software.

Existen diversas propuestas de metamodelos de proceso que se corresponden con distintos objetivos y tipos de procesos. Las más significativas son (Breton y Bézivin, 2001):

- PSL (*Process Specification Language*) (Schlenoff et al, 2000), que está orientado al dominio de los procesos de fabricación, es decir, aquellos cuyos productos son materiales. Este lenguaje, incluyendo su meta-modelo subyacente, es estándar NIST y está propuesto también para estándar ISO.

- PIF (*Process Interchange Format*) (Lee et al, 1998), que fue diseñado inicialmente para procesos de negocio. Actualmente está en un proceso de convergencia con PSL buscando integrar los dominios de procesos de fabricación y de negocio.
- ARIS (*Architecture of Integrated Information Systems*) (Scheer, 1999), que incluye un metamodelo de proceso de negocio industrial (finanzas, banca, administración, etc.).
- Metamodelo WfMC para procesos de flujos de trabajo (WfMC, 1995 y 1999), es decir, procesos reificados y controlados por un motor de flujos de trabajo (normalmente como componente central de un SGFT).
- SPEM (*Software Process Engineering Metamodel Specification*), que es una propuesta de metamodelo del OMG (2001b) basada en UML y MOF.

Por sus características, la propuesta SPEM hubiera sido muy útil y hubiera ahorrado bastante trabajo en el desarrollo de esta tesis si hubiera estado disponible a tiempo. Desgraciadamente se hizo público un borrador en diciembre de 2001 (del cual tuvo noticias este autor en febrero de 2002), bastantes meses después de las propuestas de metamodelos del Entorno MANTIS (Ruiz et al, 2001a y 2001b). Aunque la propuesta formal no está aprobada todavía (de hecho, en estos momentos no está disponible ningún documento en la web oficial) y no se incluyen aspectos de reificación de procesos ni algunos otros de los incluidos en MANTIS, hemos considerado oportuno incluir en esta tesis el anexo F con una descripción general de sus características. Esta propuesta tiene un inconveniente frente a la adoptada en MANTIS debido a que el metamodelo SPEM está definido a partir del metamodelo de UML, lo cual introduce una complejidad bastante mayor que se puede comprobar a simple vista comparando ambos metamodelos. Aun así, el estudio de la adopción de SPEM (complementado con algún otro metamodelo) en el Entorno MANTIS queda como una propuesta de trabajo futuro buscando las ventajas evidentes de la estandarización.

Debido a la diversidad de intereses y características a la hora de modelar procesos, se han propuesto arquitecturas de metamodelos basadas en los diferentes dominios de aplicación (Figura 5-25), de forma que todos los metamodelos se pudieran definir a partir de un metamodelo genérico de proceso.

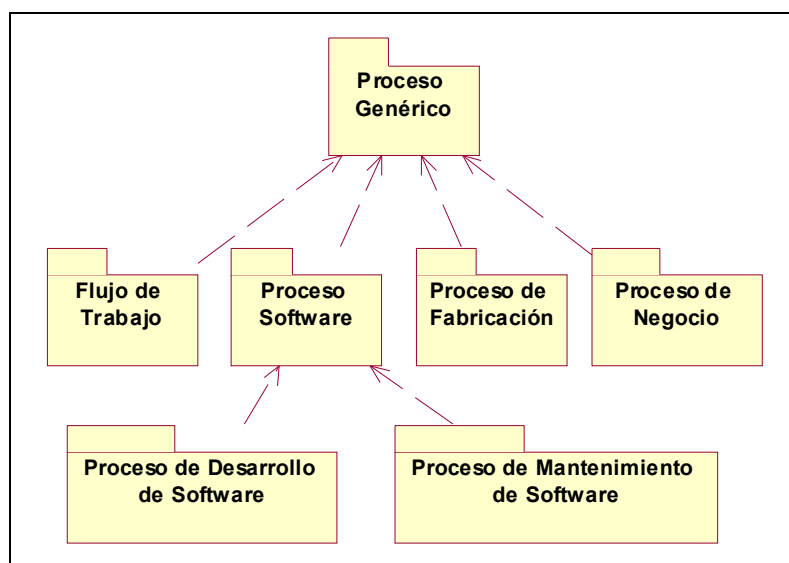


Figura 5-25. Arquitectura de metamodelos basada en dominios.

No existe actualmente ningún estándar de metamodelo genérico de proceso, aunque parece claro que debería incluir lo común a todos los tipos de procesos: ¿qué se hace? (actividades); ¿quién lo hace? (agentes); y ¿qué se obtiene? (productos). En este sentido, en Breton y Bézivin (2000) se dispone de un estudio comparativo de las principales propuestas citadas anteriormente en cuanto a su tratamiento de los tres conceptos nucleares: actividad, agente y producto.

En función de las características y objetivos del Entorno MANTIS, el metamodelo de proceso software definido integra también aspectos del dominio de proceso de negocio. Además, por las mismas razones comentadas al incluir la ontología correspondiente (apartado 5.5.4), también hemos incorporado características de flujos de trabajo. La utilidad de integrar metamodelos de FT's para los aspectos de control de la ejecución de procesos software ha sido confirmada muy recientemente por Breton y Bézivin (2002). Estos autores establecen una distinción conceptual entre definición de procesos y ejecución de procesos, de forma que un metamodelo de proceso tendría la “definición del proceso” (parte estática que no cambia con el tiempo) y “ejecución del proceso” (parte dinámica que evoluciona con el paso del tiempo). Por ejemplo, “actividad” (una clase de acción) pertenecería a la primera parte mientras que “ocurrencia de actividad” (una acción que ocurre en un momento y lugar específicos) pertenecería a la segunda. La Figura 5-26 muestra este planteamiento de forma resumida.

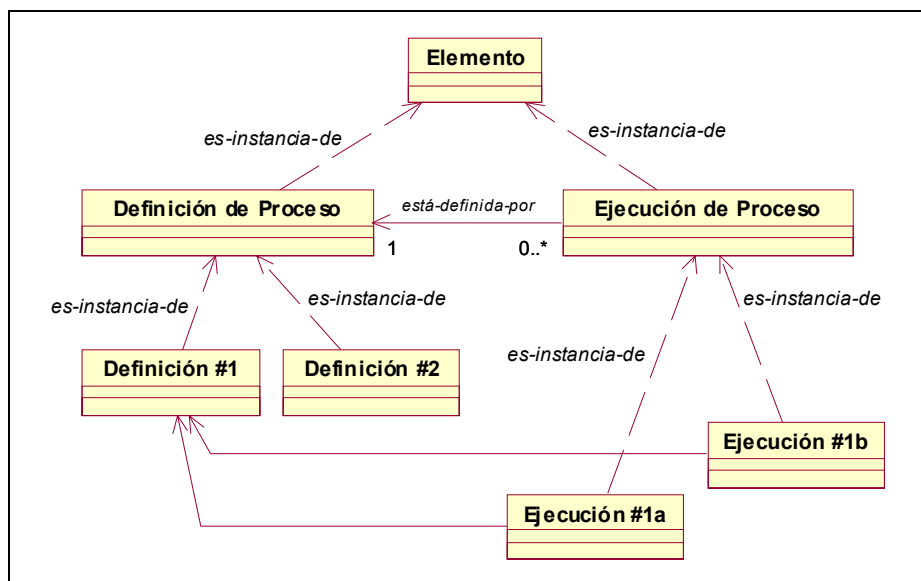


Figura 5-26. Relación entre definición y ejecución (reificación) de proceso.

Aunque el planteamiento de estos autores es aplicable al Entorno MANTIS, y por tanto se debería incluir un “metamodelo de ejecución de procesos”, no ha sido necesario hacerlo porque en MANTIS la responsabilidad de reificación de los procesos se delega en una herramienta externa de tipo SGFT, que desempeña el rol del motor de procesos de la arquitectura PSEE (ver capítulo 3).

Los documentos de base utilizados para la elaboración del metamodelo general de procesos software de MANTIS, además de las ontologías presentadas en apartados anteriores, han sido los siguientes:

- El esquema conceptual para modelado de proceso software de Becker-Kornstaedt y Webby (1999);
- El metamodelo de FT's de la WfMC (WfMC, 1995 y 1999).
- El metamodelo de flujos de trabajo de Liu et al (1999).

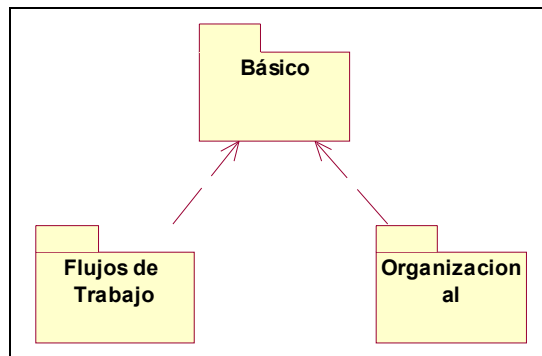


Figura 5-27. Paquetes que forman el metamodelo de proceso software.

Este metamodelo está organizado en un paquete central y dos paquetes complementarios, uno para el modelado organizacional y otro para los flujos de trabajo (Figura 5-27). A continuación se describen las características y contenido de cada uno de estos paquetes.

5.6.1.1. Paquete Básico.

Como se puede comprobar en la Figura 5-28, este paquete incluye los meta-objetos y meta-asociaciones centrales de la ontología del mantenimiento. El diferente objetivo de los metamodelos y de las ontologías implica que en los primeros no sea necesario ni adecuado incluir las diversas taxonomías que sí aparecen en las segundas. Por ejemplo, toda la jerarquía de tipos de actividades de la sub-ontología de las actividades se puede representar, y así aparece en el metamodelo, como un único meta-objeto “Actividad”.

Las principales características de este paquete, tratando de no repetir lo que es igual que en las ontologías, son las siguientes:

- No es necesario definir meta-objetos de tipo asociación o interrelación porque para ello se utilizan las asociaciones-MOF como constructores.
- Los *tipos de restricciones* son tres: objetivos (post-condiciones), pre-requisitos o pre-condiciones, e invariantes (que deben cumplirse durante todo el ciclo de vida del proyecto). No se definen restricciones a nivel de actividad o sub-actividad porque la gestión de restricciones (en tiempos, costes, recursos, etc.) se realiza siempre a nivel de proyecto o sub-proyecto, ya que esto es precisamente lo que distingue estos conceptos de otras definiciones de trabajo como proceso o actividad (PMI, 2000). Por ejemplo, el coste de una actividad es una propiedad de ejecución de la actividad pero no tiene sentido imponerlo como restricción (de obligatorio cumplimiento) ya que no existe problema en sobrepasar

dicho coste siempre y cuando no se supere el total del proyecto o sub-proyecto correspondiente. El único tipo de restricciones que interesan a nivel de actividad son las post-condiciones que definen los artefactos producidos al concluir la actividad, pero ya están incluidas directamente en el metamodelo.

- Los *artefactos entregables* son aquellos que han sido definidos formalmente como tales en el alcance del proyecto.
- Los parámetros utilizados para la ejecución de actividades en algunos metamodelos se representan mediante artefactos que son entradas de actividades.
- El *nivel de abstracción de una actividad* indica la jerarquía en el sistema de procesos utilizado: categoría de procesos, proceso, grupo de actividades, actividad, sub-actividad, etc. Se ha optado por mantener un único concepto “Actividad” para representar todos los niveles de abstracción de las definiciones de trabajo. Creemos que esta opción es mejor porque aporta mayor homogeneidad conceptual y sencillez al metamodelo. La mayoría de las otras propuestas de metamodelos distinguen meta-objetos diferentes para más de un nivel; por ejemplo, el metamodelo de la WfMC distingue entre “Proceso”, “Actividad” y “Paso” (sub-actividad atómica).
- Los *procedimientos* pueden ser de diferentes tipos: técnicas, métodos, guías, manuales de uso de herramientas, plantillas de documentos, etc.

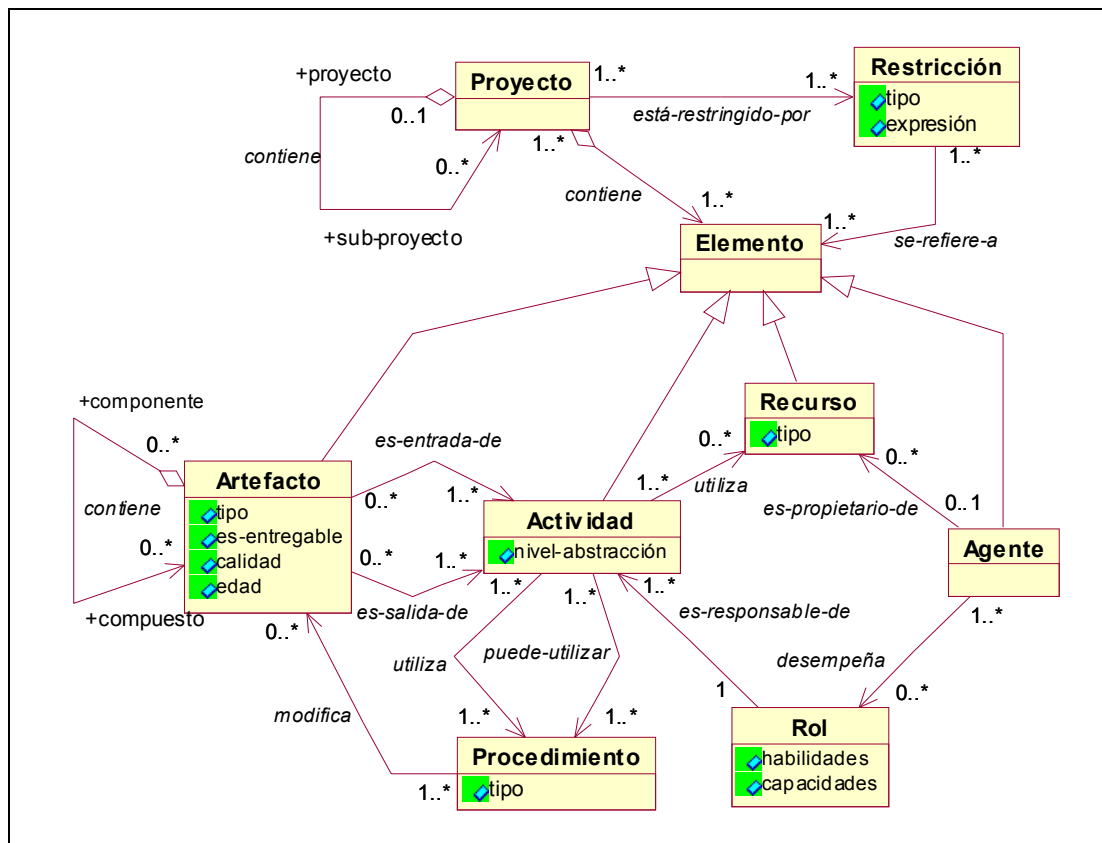


Figura 5-28. Metamodelo genérico de proceso software: paquete básico.

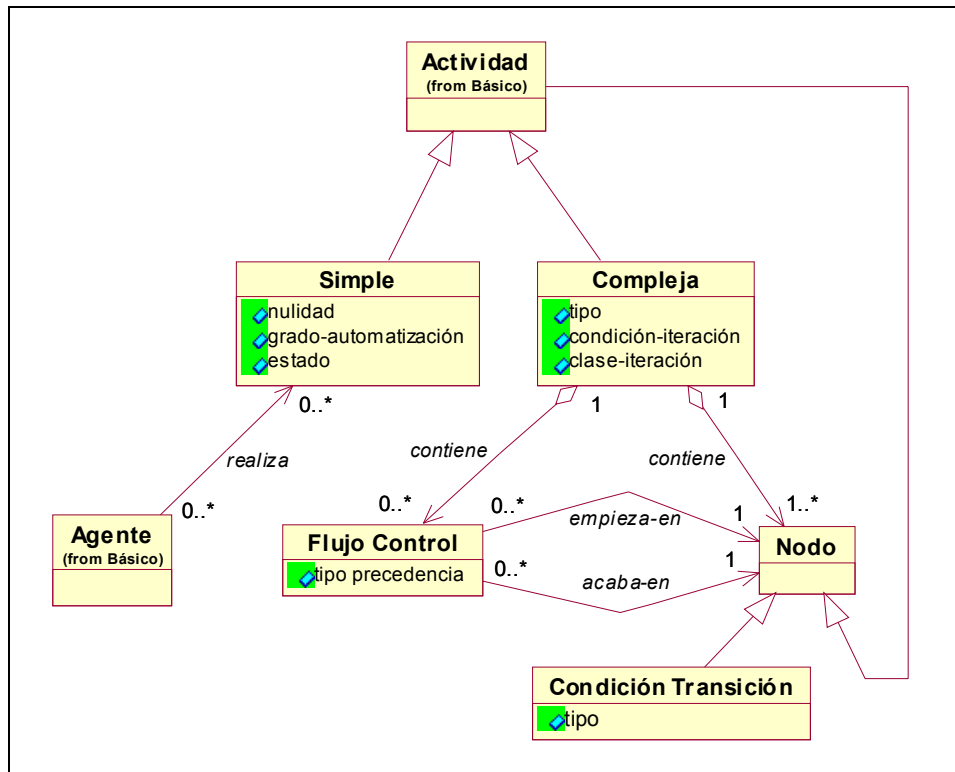


Figura 5-29. Metamodelo genérico de proceso software: paquete de flujos de trabajo.

5.6.1.2. Paquete de los Flujos de Trabajo.

El paquete de los flujos de trabajo permite extender el paquete básico con los aspectos de estructura interna y realización de actividades (Figura 5-29), basándose en la ontología de los flujos de trabajo del Entorno MANTIS. Sus principales características son:

- Las *actividades simples* (o atómicas) pueden ser *actividades nulas* (*nulidad=true*), es decir, que no definen ningún trabajo real. El único interés de utilizar estas actividades nulas es por razones de diseño del diagrama de flujos de trabajo. Por ejemplo, en la Figura 5-18 es necesaria la actividad 2, aunque no exista realmente, para representar la concurrencia. También reciben el nombre de actividades “*route*” por su utilidad para representar “*rutas*” alternativas síncronas o asíncronas.
- En función de su estructura interna, las *actividades complejas* pueden ser anidadas o iteradas. Las *actividades anidadas* tienen una estructura interna representada por un FT (colección de nodos interrelacionados mediante flujos de control) similar al de la Figura 5-18. Las *iteraciones* o bucles representan una actividad anidada incluida en un bucle de ejecución que se repetirá según se cumpla una *condición de iteración*. Las iteraciones pueden ser de dos clases: “*while*” o “*repeat-until*”.
- Las *condiciones de transición* pueden ser de dos tipos: “*or-split*” cuando el número de flujos de control que acaban en ella es 1 y el de los que empiezan es mayor de 1; y “*or-join*” cuando el número de flujos de control que acaban en ella es mayor de 1 y el de los que empiezan es 1.
- El control del *estado de ejecución de las actividades* (apartado 5.5.4.2) se realiza a nivel de actividades simples. El estado de ejecución de una actividad compleja está definido por los estados de ejecución de las sub-actividades que engloba.

- El control de la *realización de las actividades* por parte de los agentes (humanos o software) se realiza a nivel de actividades simples. Por tanto, las actividades simples son la porción de trabajo más pequeña a efectos de gestión, es decir, planificación, ejecución y control del proyecto. En términos del modelo PMBOK de gestión de proyectos (PMI, 2000), las actividades atómicas equivalen a los paquetes de trabajo (*work package*). En el caso de agentes software, la asociación “realiza” representa la invocación de aplicaciones software para la realización automática de la actividad de forma total o parcial (esto último se define mediante el atributo *grado de automatización*).
- Las *relaciones de precedencia* entre dos actividades definidas por los *flujos de control* pueden ser de tres tipos diferentes: “comenzar-comenzar”, “acabar-comenzar” y “acabar-acabar”. No se permiten dependencias “comenzar-acabar” porque violarían el modelo de FT’s estructurados utilizado.

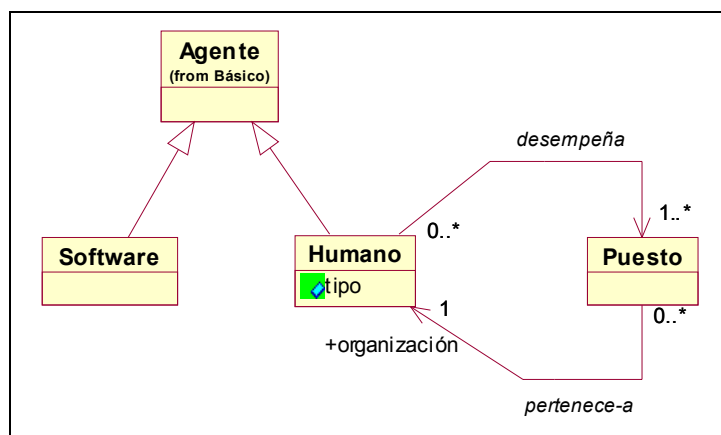


Figura 5-30. Metamodelo genérico de proceso software: paquete organizacional.

5.6.1.3. Paquete Organizacional.

Aunque existen propuestas de metamodelos organizacionales bastante complejas, se ha optado por diseñar un paquete muy sencillo (Figura 5-30) pero que permite representar cualquier posible situación, en base a las siguientes consideraciones:

- Los agentes humanos (personas u organizaciones) desempeñan ciertos *puestos* (de trabajo).
- Cada puesto puede ser desempeñado por varios agentes humanos (personas u organizaciones) y pertenece a un único agente humano (de tipo organización).

5.6.2. Metamodelo de la Medida.

El metamodelo de la medida está basado en la ontología de la medida presentada anteriormente (apartado 5.5.5). Por otro lado, tal como se mostró en la Figura 5-24, este metamodelo extiende el metamodelo de proceso software ya presentado. Además, puesto que el proceso de medición del software es un proceso software, le son de aplicación directa los conceptos incluidos en el metamodelo de proceso software. En conclusión, el metamodelo de la medida consta de un único paquete (ver Figura 5-31) en el que se incluyen aquellos aspectos de la ontología de la medida no contemplados en el metamodelo de proceso software.

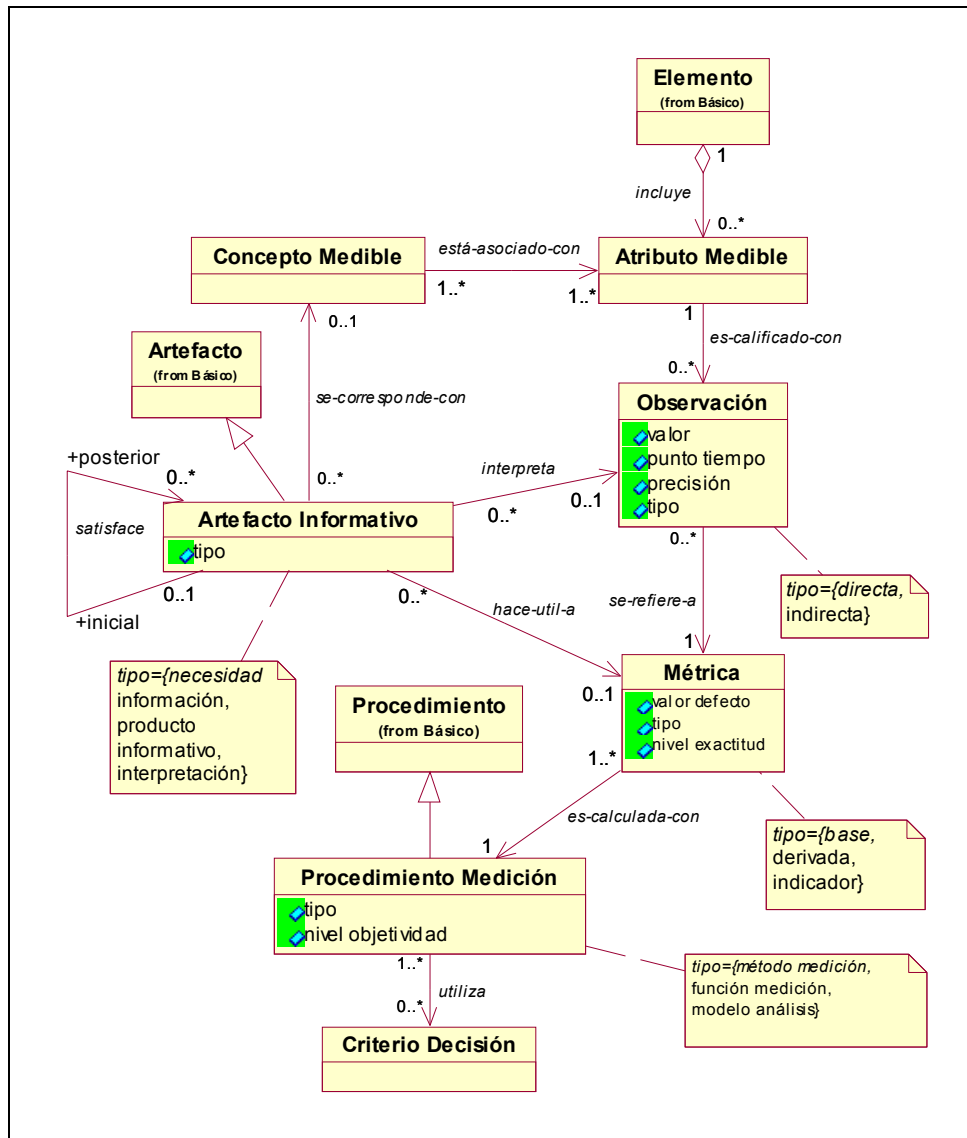


Figura 5-31. Paquete del metamodelo de la medida.

Las consideraciones para decidir cuales son los aspectos de la ontología de la medida que se pueden representar con el paquete básico de proceso software y cuales son las que es necesario incluir en el paquete de la medida, son las siguientes:

- Son tipos especiales de *artefactos* los siguientes: necesidades de información (entradas de las actividades de medición), productos informativos (salidas de las actividades de medición) e interpretaciones (sub-artefactos que forman parte de los productos informativos).
- Son *procedimientos* utilizables por las actividades de medición los siguientes: métodos de medición (para las métricas base), funciones de medición (para las métricas derivadas) y modelos de análisis (para los indicadores). Algunos procedimientos de medición (los modelos de análisis) pueden tener asociados criterios de decisión. Estos últimos no están contemplados en el paquete básico por lo que deben incluirse en el paquete de la medida.
- Una *métrica* no se considera un recurso porque, aunque lo podría parecer según la definición ontológica que hemos realizado de recurso (ver Tabla 5-9), no se trata de

elementos de un proyecto sino de algo externo a la idea de proyecto. Si se considerara que métrica es un tipo de recurso se cometería el mismo error que al considerar un algoritmo como un recurso de un proyecto cuando, en realidad, los recursos son las herramientas software que utilizan dicho algoritmo o los productos (módulos o componentes) que lo implementan. Una métrica tampoco es un procedimiento ya que en la ontología de MANTIS se distingue entre la métrica o medida (una variable) y el procedimiento para realizar observaciones y obtener su valor.

- Una *observación* en la ontología de MANTIS no es una actividad simple sino una de las múltiples acciones elementales (como llamar por teléfono, hacer una pregunta a un compañero, etc.) que no son contempladas dentro de la jerarquía de actividades de interés para la gestión de un proyecto. Una observación tampoco es un artefacto a efectos de gestión de un proyecto, aunque los resultados de una observación se pueden incluir dentro de artefactos, por ejemplo, dentro de una interpretación o dentro de un informe de pruebas.
- Los atributos medibles no se pueden englobar en las categorías de elementos de proyecto contempladas en el metamodelo de proceso (artefactos, actividades, recursos y agentes). Tampoco son procedimientos, roles, restricciones o información sobre el flujo de control del proyecto. En consecuencia, los atributos medibles deben incluirse en el paquete de la medida. Lo mismo puede decirse de los demás conceptos que son específicos del trabajo de medir: concepto medible, unidad de medición y escala.

*“No hay problemas resueltos. Sólo más o menos resueltos”
(Henry Poincaré, 1854-1912).*

6. Soporte a la Gestión del Mantenimiento

En este capítulo se presentan las herramientas metodológicas del Entorno MANTIS. La estructura del capítulo está basada en el sistema de procesos de MANTIS (capítulo 5). De esta manera, se han establecido dos categorías de métodos: para procesos de gestión, y para procesos de la organización. A su vez, cada una de ellas está organizada en base a los procesos correspondientes. Obviamente, no se ha tratado de elaborar un catálogo completo de métodos útiles para la gestión del mantenimiento del software, para lo cual ya se incluyó un estado del arte en el capítulo 4, sino de presentar, de forma no excesivamente extensa y tediosa, la colección de procedimientos específicos, o adaptados, desarrollados dentro del alcance del trabajo de esta tesis. La metodología MANTEMA no se incluye por las razones ya comentadas en el capítulo 5. Tampoco se abordan aspectos de implementación software de estos métodos y técnicas, que quedan para el capítulo 7 dedicado a las herramientas tecnológicas, es decir, las aplicaciones software desarrolladas.

6.1. Introducción.

En el capítulo 5 se han presentado las características y componentes principales del Entorno MANTIS y el marco de trabajo conceptual propuesto. A continuación se presentan las herramientas de carácter metodológico. Este grupo de elementos de MANTIS está formado por una colección de procedimientos desarrollados especialmente para el PMS teniendo en cuenta sus características y necesidades especiales (Piattini et al, 2000). Todas las propuestas presentadas en este capítulo se corresponden con el nivel M1 de la arquitectura conceptual de MANTIS (capítulo 5). En concreto, cada una de ellas, se considera un procedimiento (ver sub-ontología de organización del proceso en el capítulo 5) de alguno de los tipos definidos en las ontologías de MANTIS: métodos, técnicas o guías.

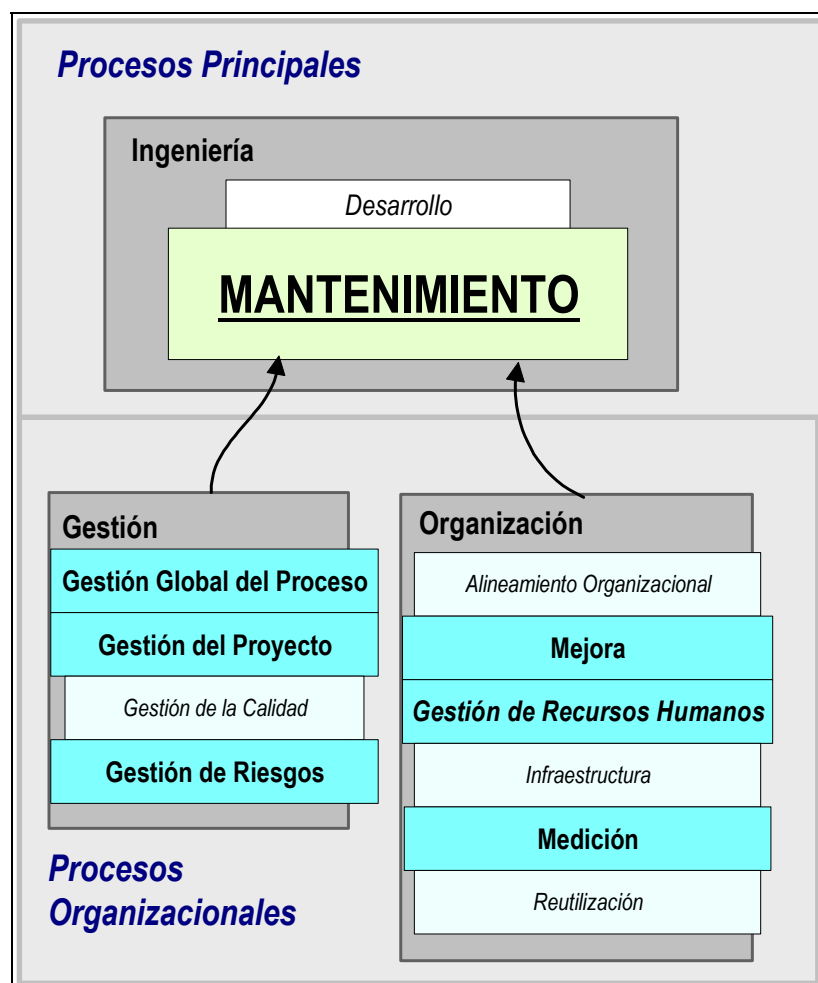
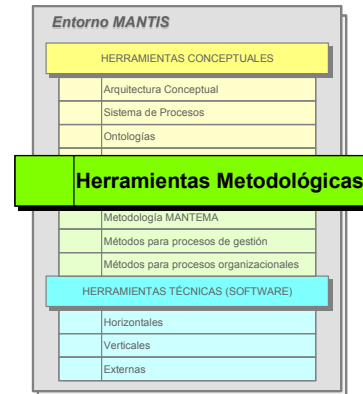


Figura 6-1. Procesos organizacionales incluidos en MANTIS.

Cada propuesta aporta uno o varios procedimientos que pueden utilizarse para realizar actividades de gestión de proyectos de mantenimiento de software, entendiendo por tales las incluidas en los procesos del subsistema organizacional del sistema de procesos de MANTIS (capítulo 5). Al igual que en éste último, las propuestas se han clasificado en dos categorías: de gestión y de la organización. En la Figura 6-1 se muestran resaltados los procesos para los cuales se han desarrollado propuestas específicas incluidas en este capítulo. Todas ellas intentan cubrir necesidades no satisfechas por la bibliografía conocida (ver capítulo 4) o, en algunos casos, realizan adaptaciones de otros procedimientos orientadas al proceso de desarrollo.

Algunas propuestas complejas también implican un modelo de actividad que se representa en MANTIS mediante el flujo de trabajo asociado a dicha actividad (ver sub-ontología de los flujos de trabajo en capítulo 5). A título de ejemplo, se ha elegido el proceso de mejora (apartado 6.3.1) para mostrar cómo se lleva a cabo la incorporación de estos procedimientos en el Entorno MANTIS, incluyendo la representación de los diversos modelos y metamodelos (niveles M1 y M2) y sus correspondencias respectivas.

Los procesos de soporte (documentación, verificación, validación, etc.) no están incluidos en el alcance del Entorno MANTIS, si bien son utilizados indirectamente porque los procesos de gestión y organización necesitan recurrir a ellos. Además, en cada actividad y tarea de la metodología MANTEMA se especifican los interfaces con los procesos de soporte (ver anexo B).

En cuanto al método de trabajo, y tal como se comentó en el capítulo 2, los métodos de investigación cuantitativa sólo han sido utilizados de forma puntual en el desarrollo de esta tesis. En esos pocos casos, por ejemplo, para validar empíricamente los factores de riesgo en proyectos de mantenimiento, se han seguido las directrices publicadas por Kitchenham et al (2001b), útiles tanto para estudios de casos, encuestas y experimentos controlados y no controlados. Estas directrices son 36 recomendaciones de qué hacer y qué no hacer en seis tópicos básicos: contexto experimental, diseño experimental, conducción del experimento y recogida de datos, análisis, presentación de resultados, e interpretación de dichos resultados.

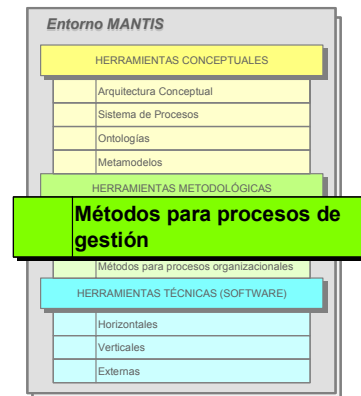
6.1.1. Gestión de la Complejidad del Mantenimiento.

Los procesos software, y en particular el mantenimiento, son inherentemente complejos. Estos procesos involucran a bastantes personas diferentes, cada una con responsabilidades y habilidades personales, y producen o modifican un amplio abanico de artefactos intermedios o entregables (Becker-Kornstaedt y Webby, 1999). El marco conceptual del Entorno MANTIS, presentado en el capítulo 5, es en sí mismo una ayuda para gestionar esta complejidad innata a los proyectos de mantenimiento de software (Ruiz et al, 2001b; Márquez et al, 2001a; García et al, 2001a). Así, las ontologías nos ayudan a comprender y describir el dominio del problema y a compartir con los demás (y con el sistema) el mismo “lenguaje” definiendo un vocabulario común. La arquitectura conceptual facilita también dicha comprensión permitiendo elevar el nivel de abstracción en el cual pensamos y diseñamos y proveyendo la encapsulación necesaria para trabajar más fácilmente con grandes cantidades de información. A su vez, el sistema de procesos de MANTIS nos permite enmarcar y analizar el dominio del problema desde una perspectiva orientada a procesos que tiene bastantes ventajas a la hora de abordar la complejidad (ver capítulo 3). Igualmente, los metamodelos (y su representación en el sistema mediante estándares de intercambio de datos y metadatos) permiten diseñar y construir un sistema más flexible adaptable a diferentes organizaciones, modelos de procesos y proyectos. Estos metamodelos son muy útiles para el intercambio de información entre los diversos artefactos

software y para una mayor interoperabilidad entre herramientas, aplicaciones, repositorios y software externo al sistema. También facilitan el diseño y construcción de herramientas mucho más parametrizadas y, por tanto, aplicables a más situaciones (ver capítulo 7).

6.2. Métodos para los Procesos de Gestión.

En esta parte se presentan los métodos y técnicas definidos para los procesos de gestión global del proceso (de mantenimiento), de gestión del proyecto, y de gestión de riesgos. No se ha desarrollado ningún método específico para el proceso de gestión de la calidad dentro del alcance del Entorno MANTIS.



6.2.1. Gestión Global del Proceso.

En el capítulo 4 se ha incluido un estado del arte de las propuestas metodológicas para abordar la gestión del PMS. Entre ellas, figura la metodología MANTEMA desarrollada por los miembros del Grupo Alarcos de I+D, al cual pertenece el autor de esta tesis. La versión 2 de MANTEMA (anexo B) fue la principal contribución de la tesis doctoral de Macario Polo (2000). La versión 3 de MANTEMA ha sido adaptada para contemplar el suministro de servicios de mantenimiento de software (externalización). Estos aspectos se presentan a continuación. También se presenta la incorporación de la gestión de conocimiento para la mejora del PMS, aunque este aspecto no está cerrado y es una de las líneas de trabajo abiertas actualmente en el Entorno MANTIS. Otra línea nueva que se pretende abordar en el ámbito del entorno MANTIS es desarrollar una versión ágil de MANTEMA, intentando trasladar al PMS ideas y técnicas propuestas por los métodos ágiles (XP y similares). Para ello, se ha presentado el proyecto MAS (Mantenimiento Ágil del Software) a la última convocatoria oficial del Ministerio de Ciencia y Tecnología (ver capítulo 8). Entre otros objetivos, en este proyecto se pretenden trasladar a MANTEMA (y al Entorno MANTIS en general) las ideas de Poole et al (2001) y de Fioravanti (2002) comentadas en el estado del arte (capítulo 4).

6.2.1.1. Externalización de Servicios de Mantenimiento.

En la actualidad la externalización u “*outsourcing*” del mantenimiento de software es uno de los sub-sectores, dentro del sector económico de servicios informáticos, que más deprisa está creciendo. Esta positiva evolución es debida a las ventajas que aporta a las organizaciones que reciben el servicio (Piattini et al, 2000):

- Un mayor control sobre el proceso gracias a que las intervenciones de mantenimiento se planifican.
- Reducción progresiva del mantenimiento correctivo (debido a errores software) ya que el prestador del servicio se compromete a alcanzar dicha meta al final del periodo del servicio. Para ello, el prestador debe llevar a cabo intervenciones de mantenimiento preventivo que mejoren la calidad del producto software mantenido.

Estas ventajas suelen ser superiores en muchas situaciones a los inconvenientes que implica (pérdida de control, menor conocimiento del producto software, o problemas con el personal, ...).

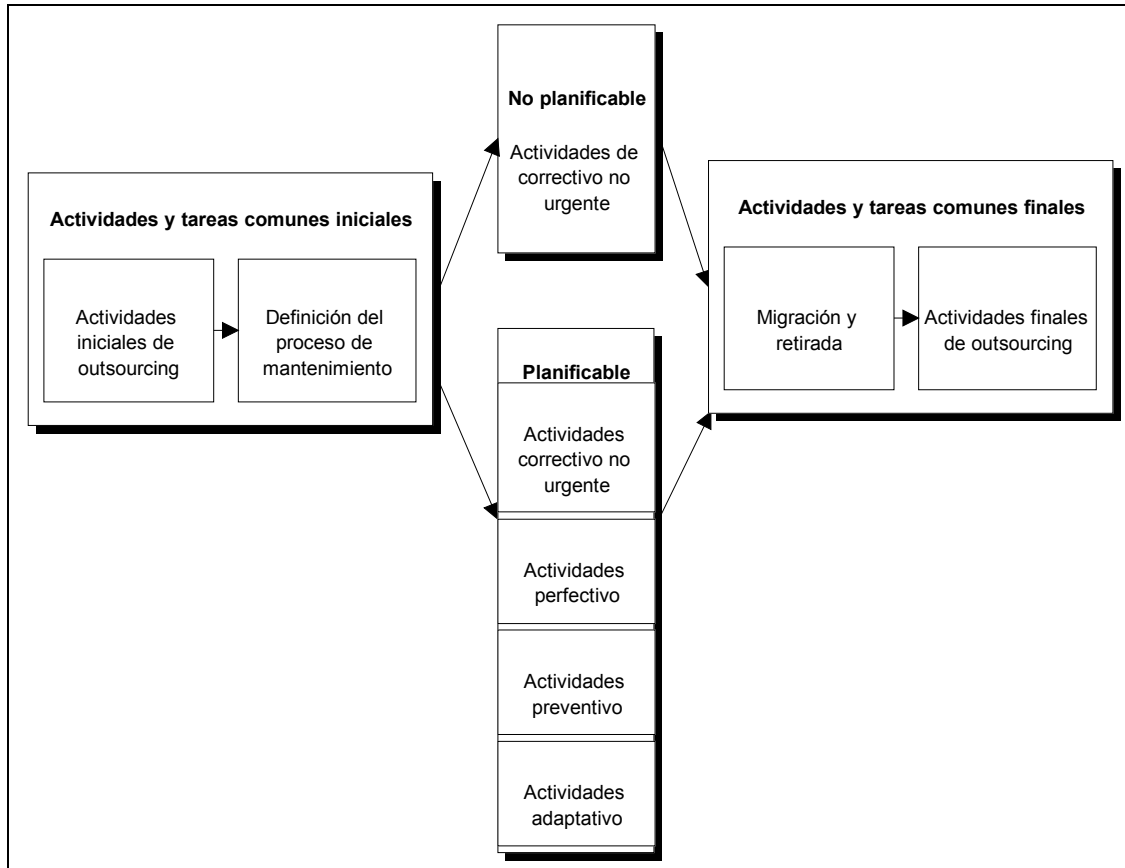


Figura 6-2. La externalización del mantenimiento en la metodología MANTEMA.

6.2.1.1.1. Externalización en MANTEMA 3.

En MANTEMA 3 se han incluido, entre otros añadidos menores, la consideración de la existencia de una organización mantenedor que presta un servicio externalizado de mantenimiento de software a una organización cliente (Polo et al, 2002b). Para ello, MANTEMA 3 establece el principio y final de la relación de externalización (*outsourcing*) entre el mantenedor y el cliente, identificando y definiendo dos conjuntos de actividades (Figura 6-2):

- Actividades iniciales de *outsourcing*, incluidas dentro del grupo de “Actividades y tareas iniciales comunes”. Son las tareas que están enfocadas a la preparación de la relación de externalización.
- Actividades finales de *outsourcing*, incluidas dentro del grupo de “Actividades y tareas finales comunes”. Son las actividades realizadas al terminar el servicio de externalización, independientemente de su tipo.

Actividad =>	I1 Estudio Inicial			I2 Implementación del Proceso		
Tarea =>	I1.1 Comienzo y recopilación de información	I1.2 Preparar propuesta de mantenimiento	I1.3 Firma del contrato	I2.1 Planificación de las relaciones cliente-mantenedor	I2.2 Adquisición de conocimiento	I2.3 Desarrollar plan
Entradas	Solicitud de servicio de mantenimiento	Cuestionario inicial Resultados de entrevistas	Propuesta de mantenimiento	Contrato de mantenimiento	Producto software operativo	Producto software a mantener
Salidas	Cuestionario inicial	Propuesta de mantenimiento	Contrato de mantenimiento	Plan de relaciones	Producto software a mantener	Plan de mantenimiento
Responsable	Cliente Mantenedor	Mantenedor	Cliente Mantenedor	Cliente Mantenedor	Mantenedor	Mantenedor
Interfaces con otros procesos				Adaptación (<i>tailoring</i>)		

Tabla 6-1. Actividades y tareas iniciales comunes en MANTEMA 3 (primera parte).

Actividad =>	I2 Implementación del Proceso (continuación)			I3 Estudio de PM	
Tarea =>	I2.4 Definir procedimientos para PM's	I2.5 Implementar proceso de GCS	I2.6 Preparar entornos de pruebas	I3.1 Recepción de PM	I3.2 Decidir tipo de intervención
Entradas	Plan de mantenimiento	Producto software a mantener Plan de mantenimiento	Producto software a mantener	Petición de Modificación (PM)	PM recibida
Salidas	Modelos de documentos Normas reguladoras	Proceso GCS	Copias de los elementos software a mantener	PM recibida	Informe de intervención
Responsable	Mantenedor	Mantenedor	Mantenedor	Mantenedor	Mantenedor
Interfaces con otros procesos		Gestión de Configuración software (GCS)	Aseguramiento de Calidad		

Tabla 6-2. Actividades y tareas iniciales comunes en MANTEMA 3 (segunda parte).

6.2.1.1.1. Actividades y Tareas Iniciales Comunes.

La lista de actividades y tareas definidas en este grupo se muestra en la Tabla 6-1 y en la Tabla 6-2 (que es continuación de la anterior). El objetivo de la primera actividad (estudio inicial) es establecer las bases para la relación de externalización entre el cliente y el mantenedor. Una vez el contrato de mantenimiento está firmado, la segunda actividad (implementación del proceso) prepara y el PMS propiamente dicho, pero sin incluir todavía la gestión de las peticiones de modificación (PM). La recepción y análisis de las peticiones de modificación remitidas por el cliente al mantenedor son el objetivo de la tercera actividad (estudio de petición de modificación). A continuación se describen con más detalle cada una de dichas actividades y tareas.

La actividad de “estudio inicial” (I1) se realiza cuando el cliente ha contactado con el mantenedor para solicitarle la prestación del servicio de externalización de alguno de sus productos software. Las tareas que la forman son las tres siguientes:

- *Comienzo y recopilación de información* (I1.1): Esta tarea se “dispara” en respuesta a una petición de servicio de mantenimiento” realizada por el cliente. El equipo de mantenimiento (ver roles de MANTEMA en el anexo B), dirigido por el responsable de mantenimiento, completa el documento llamado “cuestionario inicial”, con diversos detalles del software a mantener.
- *Preparar propuesta de mantenimiento* (I1.2): Con los datos recopilados durante la tarea anterior, el responsable de mantenimiento prepara el documento llamado “propuesta de mantenimiento”, que incluye las bases para el futuro contrato de mantenimiento. Para ello, el mantenedor tiene que realizar estimaciones de los costes del mantenimiento utilizando los datos del cuestionario inicial como base de los cálculos. Para ayudar a realizar dichas estimaciones, MANTEMA 3 incluye varias técnicas nuevas:
 - o un método para identificar y estimar los riesgos del PMS durante las etapas iniciales (ver apartado 6.2.3.2);
 - o una nueva propuesta de “acuerdos de nivel de servicio” que ayudan a conocer mejor los niveles de calidad del servicio garantizados por el mantenedor, y facilitan el control posterior de dicha calidad (ver apartado 0); y
 - o un método para estimar la cantidad de recursos humanos que se deberán asignar para el mantenimiento no planificable, es decir, para el correctivo urgente (ver apartado 6.2.2.1.1).
- *Firma del contrato* (I1.3): Una vez la propuesta de mantenimiento es analizada por ambas partes, un nuevo documento llamado “contrato de mantenimiento” debe ser redactado y firmado por ambas partes.

Aunque al comenzar la actividad de “implementación del proceso” (I2) ya está firmado el contrato de mantenimiento, el mantenedor no es todavía responsable de realizar las intervenciones de mantenimiento. Tal responsabilidad sólo le será asignada a la finalización de esta actividad. Las seis tareas que forman esta actividad son las siguientes:

- *Planificación de las relaciones cliente-mantenedor* (I2.1): En esta tarea se establece un calendario detallado de las futuras reuniones conjuntas. El propósito de dichas reuniones será que el cliente pueda conocer el estado de su cartera de aplicaciones mantenidas y, opcionalmente, saber cómo se van alcanzando los hitos previstos por el mantenedor. Es necesario identificar los roles y personas que representarán a cada organización. Todo lo anterior queda reflejado en un documento llamado “plan de relaciones”.
- *Adquisición de conocimiento* (I2.2): El equipo de mantenimiento (uno de los roles de la organización mantenedor) estudia la documentación existente sobre el software que será mantenido, incluyendo código fuente, esquemas de bases de datos, etc. También realizan entrevistas con los usuarios y prestará atención a la manera actual de trabajar la organización tercera que, en caso de existir, ha realizado hasta ahora el mantenimiento. La duración de esta tarea suele ser de uno a dos meses. A su finalización, el mantenedor deberá proveer al cliente la documentación actualizada sobre el producto software.
- *Desarrollar plan* (I2.3): El equipo de mantenimiento desarrolla el “plan de mantenimiento” y perfecciona, si ya existe, o elabora otro documento llamado “resumen técnico” del producto software a mantener. Éste último debe recoger información sobre

objetivos, funciones principales, restricciones técnicas y organizacionales, resumen de los datos obtenidos en los cuestionarios anteriores, datos históricos, técnicas y métricas para la estimación, valores de las métricas, mecanismos de control, etc.

- *Definir procedimientos para peticiones de modificación* (PM) (I2.4): El mantenedor define las plantillas y los procedimientos que el cliente deberá utilizar para presentar peticiones de modificación.
- *Implementar proceso de gestión de configuración del software* (GCS) (I2.5): El mantenedor realiza la adaptación del proceso GCS estándar a las características del servicio de mantenimiento contratado.
- *Preparar entornos de pruebas* (I2.6): el equipo de mantenimiento deberá preparar copias de los elementos software a mantener para las futuras pruebas a realizar durante las intervenciones de mantenimiento.

De todas las actividades y tareas señaladas, las que se han incorporado nuevas o han sido modificadas sustancialmente para contemplar la externalización son las tres tareas de la actividad I1, y las tareas I2.1, I2.2 e I2.3 de la actividad segunda. El resto de tareas de I2 y la actividad de “estudio de PM” (I3) son prácticamente las mismas que en MANTEMA 2. Por tanto, con cada PM que llega al mantenedor, se inicia una nueva iteración en el ciclo de mantenimiento (ver ISO 14764 en capítulo 3) formado por la actividad I3 y por las actividades intermedias correspondientes al tipo de mantenimiento asignado a la PM (Polo et al, 1999a). En este ciclo se lleva a cabo la gestión de peticiones de mantenimiento ya descrita en la sub-ontología de organización del proceso (capítulo 5).

6.2.1.1.1.2. Actividades y Tareas Finales Comunes.

En el grupo de actividades y tareas finales comunes de MANTEMA están definidas cuatro actividades: “Registrar intervención”, “Actualizar la base de datos histórica”, “retirada” y “fin de la externalización”. Esta última, añadida especialmente en MANTEMA 3 para los casos de servicios de externalización, es realizada al final del periodo del servicio de externalización. Durante su realización, el mantenedor realiza el cierre administrativo del servicio de mantenimiento. Incluye las tres tareas siguientes:

- *Inventario y entrega de la documentación* (F4.1): Dependiendo de lo establecido en el contrato de mantenimiento, el mantenedor deberá entregar al cliente todos los artefactos software generados o modificados durante la vigencia del servicio.
- *Formación y traspaso de experiencia* (F4.2): es la tarea inversa a la I2.2, es decir, ahora es el mantenedor el que transfiere sus conocimientos y lecciones aprendidas al cliente a una tercera organización diferente.
- *Cierre definitivo del servicio* (F4.3): el mantenedor concluye de manera definitiva cualquier actividad de servicio al cliente.

6.2.1.1.1.3. Documentos para la Externalización.

A continuación se presenta la estructura propuesta en MANTEMA 3 para algunos de los principales documentos entregables relacionados con el servicio de externalización: “cuestionario inicial”, “propuesta de mantenimiento”, “contrato de mantenimiento” y “resumen técnico”. Aunque el principal documento entregable es el “plan de mantenimiento”, no se incluye porque no está directamente relacionado con la externalización y en MANTEMA 3, al

igual que en la anterior versión, sigue estando formado por los apartados propuestos en la norma ISO 14764 (1998e).

Cuestionario inicial.

1. Identificación de la organización cliente (datos financieros, personal, etc.) e interlocutores autorizados
2. Entorno hardware y software
 - 2.1. Hardware
 - 2.1.1. Servidores (marca y modelo, sistema operativo, etc.)
 - 2.2. Hardware periférico
 - 2.2.1. Terminales
 - 2.2.2. Estaciones de trabajo (indicando el sistema operativo de cada una)
 - 2.2.3. Tipo de red
 - 2.2.4. Tipo de conexión con los servidores
3. Entorno software
 - 3.1. Entorno de desarrollo
 - 3.2. Sistema de ficheros
 - 3.3. Sistemas de Gestión de Bases de datos
 - 3.4. JCL Batch
4. Organización que desarrolló y/o ha mantenido el software
 - 4.1. Metodologías y técnicas de desarrollo
 - 4.2. Estándares (de denominación, programación, interfaces de usuario, etc.)
 - 4.3. Técnicas y estándares de aseguramiento de calidad
 - 4.4. Métodos de gestión de proyectos
 - 4.5. Procedimiento de puesta en explotación
 - 4.6. Procedimientos de auditoria
 - 4.7. Procedimientos de resolución de problemas
 - 4.8. Procedimientos de documentación
 - 4.9. Otros procedimientos o normas utilizadas
5. Aplicaciones (un apartado por cada producto software mantenido)
 - 5.1. Identificación (nombre, versión, fecha, etc.)
 - 5.2. Unidad organizacional responsable
 - 5.3. Otras aplicaciones relacionadas
 - 5.4. Programas por lotes (lenguajes, tamaño, antigüedad, nº de módulos, nº de intervenciones sufridas, etc.)
 - 5.5. Programas interactivos (idem anterior)
 - 5.6. Listados e informes (idem anterior)
 - 5.7. Pantallas y formularios (idem anterior)

Propuesta de mantenimiento.

1. Introducción
2. Resultados de los análisis aplicados a una muestra de programas de distinta complejidad elegidos por el mantenedor
3. Propuesta de servicio
 - 3.1. Informe técnico definiendo objetivos, límites, vínculos, responsabilidades y parámetros contractuales
 - 3.2. Para cada aplicación que quede dentro de los límites del contrato: tipos de mantenimiento contemplados y sus correspondientes “acuerdos de nivel de servicio”

- 3.3. Propuesta de contrato (similar al contrato definitivo)
- 4. Oferta económica

Contrato de mantenimiento.

- 5. Identificación de las partes
- 6. Objeto del contrato (el cliente confía al mantenedor un conjunto de prestaciones, en el campo de los servicios informáticos, que tiene como finalidad el mantenimiento del software registrado en el contexto de la propuesta técnica)
- 7. Características de la prestación del servicio
 - 7.1. Inventario de los artefactos software a mantener
 - 7.2. Estado inicial del software
 - 7.3. Condiciones de organización del trabajo del mantenedor y del cliente
 - 7.4. Condiciones de formalización de las intervenciones de mantenimiento
 - 7.5. Adaptación del plan de aseguramiento de la calidad a las necesidades del cliente
 - 7.6. Corrección de las anomalías
 - 7.7. Descripción del software aplicativo y del software estándar del cliente que es mantenido por otras organizaciones diferentes al mantenedor
 - 7.8. Relación de la documentación que deberán facilitar ambas organizaciones a la otra parte. Se deben señalar, entre otros aspectos, el formato y los plazos de entrega
 - 7.9. Modalidad de asistencia a los usuarios finales
- 8. Obligaciones del cliente
- 9. Obligaciones del mantenedor
- 10. Cláusulas de exclusión
- 11. Cláusulas de organización
- 12. Garantía de las intervenciones
 - 12.1. Recuperación de datos
 - 12.2. Definición de responsabilidades
- 13. Otras cláusulas opcionales adicionales

Resumen técnico de un producto software.

- 14. Introducción
 - 14.1. Alcance y propósito
 - 14.2. Objetivos del proyecto o servicio de mantenimiento
 - 14.2.1. Objetivos
 - 14.2.2. Funciones principales
 - 14.2.3. Restricciones técnicas y organizacionales
- 15. Estimaciones
 - 15.1. Resumen de los datos obtenidos con los cuestionarios (nº de programas, tamaño, estándares usados, ...)
 - 15.2. Datos históricos (si existen)
 - 15.3. Técnicas y métricas para la estimación
 - 15.4. Valores de las métricas
- 16. Recursos
 - 16.1. Recursos humanos
 - 16.2. Recursos técnicos (hardware y software)
 - 16.3. Otros recursos
- 17. Organización del personal (estructura organizacional)
- 18. Mecanismos de seguimiento y control
- 19. Apéndices (si los hubiera)

6.2.1.1.1.4. Acuerdos de Nivel de Servicio.

Para ofrecer un servicio de más calidad, el mantenedor debe llevar a cabo su servicio de mantenimiento de acuerdo a algunos indicadores que sean medibles de la forma más objetiva posible. Con este fin, en MANTEMA 3 se proponen diferentes “Acuerdos de Nivel de Servicio” (ANS) para cada servicio contratado (es decir, cada tipo de mantenimiento). Cada uno de estos acuerdos viene definido por una colección de indicadores de servicio y de valores límite asociados, que el mantenedor garantiza al cliente. En caso de que algún valor límite de algún indicador no sea cumplido por el mantenedor, el contrato deberá especificar las compensaciones oportunas.

Estos indicadores de servicio pueden ser los propuestos en la bibliografía (De Vogel, 1999) o los nuevos propuestos en MANTEMA. Obviamente, esta lista puede ser extendida en función de las necesidades. Algunos de los propuestos en MANTEMA 3 son:

- NMR_{CA} : Número máximo de PM's relacionadas con anomalías críticas por unidad de tiempo (por ejemplo, 20 al mes).
- NMR_{NCA} : Número máximo de PM's relacionadas con anomalías no críticas por unidad de tiempo.
- TR_{CA} : Tiempo máximo de resolución de anomalías críticas (por ejemplo, 6 horas). Sólo se utiliza para mantenimiento correctivo urgente.
- TR_{NCA} : Tiempo máximo de resolución de anomalías no críticas (por ejemplo, 2 días). Normalmente se definen cuatro valores diferentes, uno por cada tipo de mantenimiento planificable (anexo B).
- Dev_{CA} : Desviación máxima en el número de anomalías críticas en un periodo de tiempo corto. O dicho de otra forma, número máximo de PM's que pueden ser recibidas en un determinado periodo (por ejemplo, una semana) con la obligación de cumplir el indicador TR_{CA} . Por ejemplo, si $NMR_{CA}=20$, $TR_{CA}=12$ horas y $Dev_{CA}=10$, significa que el mantenedor se compromete a atender 20 PM por mes en un tiempo máximo de 12 horas, pero con un máximo semanal de 10. Para las PM's que exceden las 10 semanales no existe el compromiso de las 12 horas.
- Dev_{NCA} : Similar a Dev_{CA} , pero referido a anomalías no críticas.
- PPC (compromisos previos de progreso): aunque las intervenciones de mantenimiento preventivo no suelen estar incluidas en los contratos de mantenimiento, si es conveniente establecer un compromiso de mantenimiento preventivo progresivo. Por ejemplo, se puede establecer que al final del tiempo de vigencia del servicio el mantenedor deberá haber reducido en un 10% la media de la complejidad ciclomática de los módulos sometidos a mantenimiento; o se puede negociar que el mantenedor deberá reducir el número de errores colaterales producidos por las intervenciones de mantenimiento en un 20% cada año respecto del año anterior.

Los ANS para el resto de tipos de mantenimiento planificable (perfectivo y adaptativo) no son frecuentes ni recomendables ya que es difícil estimar a priori el esfuerzo de esta clase de intervenciones, que pueden implicar cambios muy grandes en el software. Por esta razón, en MANTEMA se recomienda que las intervenciones de dichos tipos se estudien de forma individualizada.

6.2.1.2. Gestión del Conocimiento.

Actualmente, las organizaciones consideran tan importante el capital intangible (intelectual) como el tangible. Prueba de ello es que el 62% de las medianas y grandes empresas españolas ha puesto en marcha una iniciativa de gestión del conocimiento (Diario El País, 17-marzo-2002). Son múltiples las ventajas que una buena gestión del conocimiento aporta a una organización. Algunas de las más importantes son:

- Estructurar el conocimiento
- Diseminar el conocimiento donde sea más necesario o útil.
- Generar conocimiento explícito que pasa a forma parte del capital intelectual de la empresa.
- Generar nuevo conocimiento a partir de información o conocimiento previos.
- Reutilizar soluciones y/o información previas.

Todas estas ventajas ayudan a alcanzar mejor los objetivos estratégicos de una organización, incrementando su rendimiento general y su competitividad. Por esta razón, hoy en día el conocimiento o capital intelectual es un recurso crucial para las organizaciones (Larson et al, 2001), (Gupta y Govindarajan, 2000); y por ello las organizaciones están preocupadas en encontrar técnicas y métodos que gestionen su conocimiento de una forma sistemática.

Aunque en el contexto de la ingeniería del software intervienen diferentes fuentes de conocimiento que además se encuentran en continuo cambio, hasta ahora se le ha prestado muy poca atención a la gestión del conocimiento, y su utilización en este campo ha sido mas bien escasa. Sólo muy recientemente las organizaciones de la industria y servicios del software han empezado a comprender que su éxito en el futuro depende del uso y la gestión de su conocimiento actual (Komi-Sirvio et al, 2002). Más en concreto, en el tema del mantenimiento del software las publicaciones al respecto brillan por su ausencia. Tan sólo tiene alguna relación con este asunto la propuesta de Birk et al (1998) de utilizar una base de conocimientos de proyectos de mantenimiento históricos para ayudar durante la planificación de proyectos nuevos.

En línea con las ideas de Birk, y buscando poder reutilizar el conocimiento adquirido durante la planificación y ejecución de proyectos de mantenimiento, parece muy interesante y útil ampliar el Entorno MANTIS con una base de conocimientos. La incorporación de la gestión de conocimiento a MANTIS es especialmente interesante porque:

- El PMS es muy intenso en producción y utilización de conocimiento.
- Este conocimiento no proviene sólo de los expertos o profesionales, también se obtiene del producto que debe ser mantenido, de las razones que motivaron el mantenimiento (nuevos requisitos, quejas de los usuarios, etc.), y de las lecciones aprendidas con los proyectos, procesos, metodologías y herramientas utilizadas en la organización.
- El esfuerzo más importante que se realiza cuando se llevan a cabo intervenciones de mantenimiento consiste en comprender el sistema software y sus artefactos (Piattini et al, 2000), es decir, se trata de un problema básicamente de adquisición de conocimiento.
- Entre los principales factores que impactan en la productividad de los equipos de mantenimiento se encuentra la existencia de algún miembro experto en la aplicación a

mantener, la inexperiencia de los programadores y la capacidad (basada en la experiencia) del equipo en su conjunto (Banker et al, 1991; Banker et al, 2002).

- Los recientes avances tecnológicos (como el uso extensivo de Internet) han permitido que existan grandes equipos de mantenimiento del software que, con frecuencia, se encuentran geográficamente distribuidos, lo cual aumenta las dificultades para la coordinación del trabajo, proporcionar acceso al conocimiento de los proyectos, y gestionar la experiencia recopilada en proyectos previos y hacerla disponible y reutilizable (Nebus, 2001). Es decir, para el PMS resulta nefasta la situación conocida como “la mano izquierda no sólo no conoce lo que está haciendo la derecha, sino que ni siquiera sabe que existe la mano derecha” (O'Dell y Grayson, 1998).

Sin embargo, la mayoría de las empresas que realizan mantenimiento de software no utilizan técnicas que permitan gestionar el conocimiento generado y, lo que es todavía más significativo, no cuentan con mecanismos que propicien el paso del conocimiento y experiencia de cada empleado (conocimiento tácito) a conocimiento útil para la organización (conocimiento explícito). La situación más frecuente es que el ingeniero de software que lleva a cabo las intervenciones de mantenimiento es la única persona que posee el capital intelectual, y por tanto, la organización está sometida al riesgo de perderlo cuando la persona deja su puesto de trabajo o surge algún imprevisto que hace que no esté disponible. Para evitar estos problemas, y puesto que las organizaciones no saben usar su experiencia en gran parte, Basili et al (2001a) sugieren que las organizaciones que realizan desarrollo y/o mantenimiento del software utilicen herramientas que les permitan aprender desde su experiencia y compartir dicha experiencia con todos los empleados que puedan necesitarla.

En suma, en el Entorno MANTIS se propone utilizar un Sistema de Gestión de Conocimiento (SGC) para reducir o solucionar muchos de los problemas enumerados, ayudar a la coordinación del trabajo y fomentar el aprendizaje y el desarrollo de habilidades de los participantes. Esta propuesta no está concluida y actualmente es una de las líneas abiertas de I+D en el Entorno MANTIS (ver capítulo 8). Por ello, a continuación nos limitamos a comentar los fundamentos del sistema propuesto, que está basado en un conjunto de agentes software, y en el capítulo 8 se presentan las características de la herramienta KM-MANTIS, que implementa la base de conocimientos, detallando la arquitectura de agentes utilizada.

6.2.1.2.1. Sistema Basado en Agentes.

Para el diseño y construcción del prototipo del SGC de MANTIS se ha optado por utilizar varios tipos de agentes software inteligentes agrupados en comunidades, cada una de ellas “encargada” de un tipo de conocimiento diferente (Vizcaíno et al, 2002). Los motivos de esta decisión son las siguientes características de los sistemas multiagentes:

- Son una buena manera para tratar el conocimiento (Silverman et al, 1995), (Rasmus, 1996).
- Son proactivos, es decir, actúan cuando consideran que es conveniente sin que nadie tenga que indicárselo (Russell y Norvig, 1995).
- Tienen la capacidad de aprender desde su propia experiencia.
- Pueden gestionar grandes cantidades de información y razonamiento.

Aunque no existe todavía un marco teórico común para los “sistemas multiagentes” como el considerado en este documento, se ha adoptado como tal la propuesta de la FIPA

(*Foundation for Intelligent Physical Agents*) conocida como “*FIPA Agent Software Integration Specification*” (FIPA, 2000), (Paniagua et al, 2001). Adicionalmente, los principales trabajos que han aportado ideas útiles para definir este SGC han sido los siguientes:

- El SGC PATTERNS para la gestión de procesos software. Este sistema usa tres agentes inteligentes que colaboran entre sí en la resolución de los problemas de los participantes (Ostolaza et al, 2002).
- El SGC CeBASE para coleccionar y diseminar conocimiento sobre ingeniería del software (Basili et al., 2001b). Se pretende que este conocimiento pueda ser utilizado en *eWorkshops* (reuniones científicas electrónicas), con lo cual la necesidad de realizar reuniones cara a cara disminuiría. CeBASE se basa en lo que en inglés es llamado “*Experience Factory Organization*” (Basili y Seaman, 2002). La fundamentación de este concepto es que los proyectos de desarrollo de software mejorarían su rendimiento (costes, calidad, tiempos de entrega, etc.) si se utilizara la experiencia obtenida en proyectos previos.
- La aplicación DIAMS, formada por agentes que colaboran entre sí y con los usuarios para ayudarles a encontrar conocimiento (Chen et al, 2000). Además los agentes se encargan de gestionarlo y compartirlo.
- El proyecto InfoSleuth (Perry et al, 1999), que es un sistema basado en agentes que recoge información y la analiza.

Todos estos sistemas se orientan, de forma casi total y exclusiva, a la recolección de información y su mantenimiento y dejan de lado la problemática de la generación de nuevo conocimiento.

Los agentes del SGC de MANTIS deben realizar diferentes y diversas tareas para lo cual es necesario utilizar un abanico de técnicas de inteligencia artificial: redes neuronales, algoritmos genéticos, algoritmos ID3, y reglas de producción. Las tareas más destacada de estos agentes son (Vizcaíno et al, 2002):

- Controlar la consistencia de la información y comunicar a otros agentes los cambios ocurridos;
- Predecir nuevos cambios o posibles errores;
- Sugerir soluciones a problemas;
- Ayudar en la toma de decisiones, por ejemplo, aconsejar cierto personal para la realización de determinadas actividades; y
- Estimar esfuerzos y costes.

El prototipo de SGC del Entorno MANTIS, llamado KM-MANTIS, está plenamente adaptado al marco de trabajo conceptual de MANTIS. Así, el conocimiento está organizado y definido por las ontologías, el sistema de procesos y los metamodelos ya presentados en el capítulo 5. Además, otra característica básica es que dicho conocimiento está estructurado en varios niveles de abstracción coincidentes con los 4 niveles de la arquitectura conceptual de MANTIS. En el capítulo 7 se detalla la arquitectura de agentes concreta del prototipo actual y sus roles.

6.2.2. Gestión del Proyecto.

En este apartado se engloban los métodos que sirven para identificar, establecer, coordinar y supervisar las actividades, tareas y recursos necesarios en un proyecto para producir un producto o servicio que cumpla los requisitos (norma ISO 15504, capítulo 4).

Aunque el Entorno MANTIS propone como marco metodológico básico para la gestión de los proyectos de mantenimiento la norma PMBOK del “*Project Management Institute*” (PMI, 2000), algunas características especiales de este tipo de proyectos (por ejemplo, su orientación a la prestación de un servicio en vez de a la construcción de un producto) justificaran el desarrollo de algunos métodos especiales, comentados a continuación, y herramientas (explicadas en el capítulo 7), complementarios a los conocidos para gestión de proyectos software en general.

Se ha realizado una adaptación de la norma PMBOK para evitar las inconsistencias de concepto y de nomenclatura entre dicha norma y los otros estándares utilizados en MANTIS (fundamentalmente de ISO) y las ontologías que definen el dominio de conocimiento. Los cambios originados por esta adaptación son los siguientes:

- La gestión de proyectos en PMBOK se considera formada por 39 procesos diferentes, mientras que en el sistema de procesos de MANTIS (capítulo 5), al estar basado en ISO 15504 (ISO/IEC, 1998c), la gestión de proyectos es, en su conjunto, un único proceso en el que se engloban 21 actividades. Estas actividades se corresponden con los 21 procesos de PMBOK encuadrados en los grupos de “gestión de la integración”, “gestión del alcance”, “gestión de los tiempos”, “gestión de los costes” y “gestión de las comunicaciones” (ver Tabla 6-3).
- Los grupos de procesos “gestión de la calidad”, “gestión de riesgos” y “gestión de recursos humanos” de PMBOK no se consideran en MANTIS parte de la “gestión del proyecto”, sino que se corresponden con los procesos organizacionales de igual nombre. de PMBOK (ver Tabla 6-3).
- Por último, el grupo de procesos “gestión de adquisiciones” de PMBOK cubre parcialmente los objetivos del proceso “infraestructura” en MANTIS

MANTIS (procesos) <i>[basado en ISO 15504]</i>		PMBOK (grupos de procesos)
Procesos de Gestión	Gestión del Proyecto	Gestión de la Integración (3 procesos) Gestión del Alcance (5 procesos) Gestión de los Tiempos (5 procesos) Gestión de los Costes (4 procesos) Gestión de las Comunicaciones (4 procesos)
	Gestión de la Calidad	Gestión de la Calidad (3 procesos)
	Gestión de Riesgos	Gestión de Riesgos (6 procesos)
Procesos de la Organización	Gestión de Recursos Humanos	Gestión de Recursos Humanos (3 procesos)
	Infraestructura (parcial)	Gestión de Adquisiciones (6 procesos)

Tabla 6-3. Adaptación de PMBOK al Sistema de Procesos de MANTIS.

Las fases que caracterizan el ciclo de vida de un proyecto en PMBOK se han considerado adecuadas y aplicables a los proyectos de mantenimiento de software sin ningún retoque o cambio. Por tanto, en MANTIS, un proyecto pasa por las cinco fases siguientes:

iniciación, planificación, ejecución, control y finalización. También se considera el ciclo planificación-ejecución-control de PMBOK (Figura 6-3).

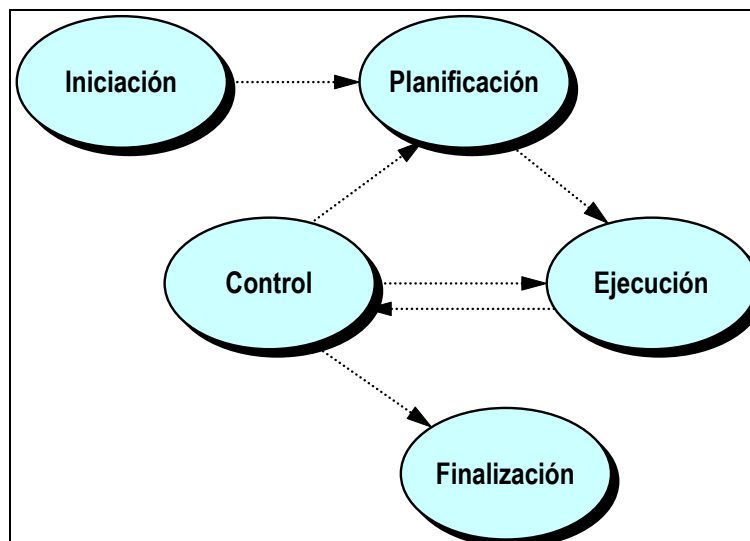


Figura 6-3. Fases de un proyecto en MANTIS.

6.2.2.1. Planificación de Recursos.

La planificación de recursos es uno de los procesos de PMBOK englobado dentro de la gestión de los costes. En el Entorno MANTIS se considera una de las actividades englobada en el proceso de “gestión del proyecto” (ver Tabla 6-3). En este apartado se presenta una técnica para la estimación de recursos humanos para mantenimiento correctivo urgente. También se incluye un caso de estudio empírico sobre la estimación del mantenimiento de aplicaciones heredadas (*legacy code*) utilizando métricas de código.

6.2.2.1.1. Recursos Humanos para el Mantenimiento Correctivo Urgente.

El mantenimiento correctivo urgente (MUC), también llamado no planificable porque sus intervenciones no tienen planificación debido a la urgencia, es uno de los cinco tipos de mantenimiento identificados en la metodología MANTEMA (anexo B). El problema con este tipo de mantenimiento es que, como su nombre indica, la urgencia que los usuarios y/o clientes tienen para resolver estas peticiones de modificación implica que las intervenciones no pueden tener actividades de planificación como en los demás tipos de mantenimiento. Sin embargo, algunos autores han modelado un ciclo de vida del PMS incluyendo este tipo de mantenimiento. Por ejemplo, Kung y Hsu (1998) observaron que la mayoría de PM de este tipo se concentran al final de la vida de cada versión de un producto software. Basili et al (1996) y Schneidewind (1998) también llegaron a conclusiones parecidas. Por otro lado, Calzolari et al (1998), Shepherd et al (1996), y Niessink y Van Vliet (1997), han propuesto diferentes métodos y herramientas (capítulo 4) para predecir el esfuerzo de mantenimiento, incluyendo el correctivo.

La técnica que presentamos a continuación complementa a las anteriores en el sentido de que, utilizando los datos obtenidos por alguno de dichos métodos de estimación de esfuerzo y los parámetros económicos del entorno del proyecto, sirve para determinar la cantidad de

recursos adecuada para evitar que el MUC origine pérdidas económicas propias al mantenedor (Polo et al, 2002b).

6.2.2.1.1.1. Modelo Económico.

Desde el punto de vista de las organizaciones cliente y mantenedor de un determinado producto software, los factores económicos a tener en cuenta son los siguientes:

- El precio (en unidades monetarias) que el cliente paga al mantenedor por cada hora-persona (h-p) de recurso contratada (μ_c).
- El coste que para el mantenedor tiene cada h-p (μ_m). Es evidente que para que el mantenedor no tenga pérdidas económicas se debe cumplir que $\mu_m \geq \mu_c$.
- El importe de la penalización que el mantenedor deja de ingresar por cada hora de retraso (s).

Si el mantenedor asigna menos recursos que los realmente necesarios, entonces incurrirá en retrasos que implicaran penalizaciones en sus ingresos. Sin embargo, el mantenedor podría estar interesado en incurrir en algunos retrasos si las penalizaciones correspondientes fuesen inferiores a los costes de los recursos que tendría que añadir. Por tanto, la pregunta correcta, desde el punto de vista del mantenedor, no es ¿cuántos recursos tengo que asignar para MUC para evitar penalizaciones?, sino ¿cuántos recursos tengo que asignar para MUC para no tener pérdidas económicas?. La técnica siguiente ayuda a responder a esta última.

Día	Horas-Persona acordadas	Horas-Persona reales	Días necesarios	Días necesarios acumulados	Retraso	Penalización
1	h	p	$\frac{h}{p}$	$\frac{h}{p}$	$\frac{h}{p} - 1 = \frac{h-p}{p}$	$s \cdot \frac{h-p}{p}$
2	h	p	$\frac{h}{p}$	$2 \cdot \frac{h}{p}$	$2 \cdot \frac{h}{p} - 2 = 2 \cdot \frac{h-p}{p}$	$2s \cdot \frac{h-p}{p}$
3	h	p	$\frac{h}{p}$	$3 \cdot \frac{h}{p}$	$3 \cdot \frac{h}{p} - 3 = 3 \cdot \frac{h-p}{p}$	$3s \cdot \frac{h-p}{p}$
.....						
n	h	p	$\frac{h}{p}$	$n \cdot \frac{h}{p}$	$n \cdot \frac{h-p}{p}$	$n \cdot s \cdot \frac{h-p}{p}$
TOTAL	n · h	n · p	$n \cdot \frac{h}{p}$	$n \cdot \frac{h}{p}$	$\left(\sum_{i=1}^n \right) \cdot \frac{h-p}{p}$	$\left(\sum_{i=1}^n \right) \cdot s \cdot \frac{h-p}{p}$

Tabla 6-4. Modelo económico diario.

Si el mantenedor y el cliente determinan en el ANS que los recursos previstos para atender el MUC cada día serán 'h' horas-persona (todos los días igual), la Tabla 6-4 muestra la situación para un periodo de 'n' días si el mantenedor, en vez 'h', realmente dedica 'p' horas-persona cada día. Cada fila muestra los valores del día indicado en la primera columna. La segunda y tercera columnas refieren los recursos diarios acordados (h) y los realmente dedicados por el mantenedor (p). El valor de h puede ser obtenido por alguna de las técnicas de estimación propuestas por diversos autores y citadas anteriormente. La cuarta columna indica los días necesarios para atender por el mantenedor lo que, según el ANS, debería ser atendido en

un día para evitar las penalizaciones, mientras que la quinta columna muestra dichos días de forma acumulativa. Como se puede observar en la fila de totales, el resultado será que el mantenedor, en vez de ‘n’ días tardará “n*(h/p)”. Por tanto, si $p < h$ se producirá un retraso y si $p > h$ se producirá un adelanto respecto de lo acordado. Las columnas seis y siete muestran, para la primera situación, el retraso en días producido y su penalización.

Con la situación descrita, los ingresos ‘I’ del mantenedor (es decir, los costes para el cliente) y sus costes ‘C’ sin penalizaciones son, respectivamente, los siguientes:

$$I = n \cdot h \cdot \mu_c \quad (6.1)$$

$$C(p) = n \cdot p \cdot \mu_m \quad (6.2)$$

mientras que las penalizaciones ‘S’ que sufre son (valor de la última columna en la fila de totales de la Tabla 6-4)

$$S(p) = \left(\sum_{i=1}^n i \right) \cdot s \cdot \frac{h-p}{p} = \frac{n \cdot (n+1)}{2} \cdot s \cdot \frac{h-p}{p} = \frac{n \cdot (n+1) \cdot s \cdot (h-p)}{2 \cdot p} \quad (6.3)$$

Los beneficios ‘B’ del mantenedor son los ingresos menos los costes y menos las penalizaciones:

$$B(p) = I - C(p) - S(p) \quad (6.4)$$

y por tanto, para que el mantenedor no sufra pérdidas económicas se debe cumplir que $B(p) > 0$, es decir:

$$I \geq C(p) + S(p) \quad (6.5)$$

o, lo que es lo mismo

$$(n \cdot h \cdot \mu_c) \geq \left[(n \cdot p \cdot \mu_m) + \left(\frac{n \cdot (n+1) \cdot s \cdot (h-p)}{2 \cdot p} \right) \right] \quad (6.6)$$

Eliminando ‘n’ y despejando respecto de ‘p’ se obtiene la siguiente inecuación de segundo grado, útil para calcular p:

$$(2\mu_m) \cdot p^2 - (s \cdot (n+1) + 2h \cdot \mu_c) \cdot p + (s \cdot h \cdot (n+1)) \leq 0 \quad (6.7)$$

6.2.2.1.1.2. Ejemplo y Comentarios.

En la Tabla 6-5 se muestran los resultados obtenidos con esta técnica para el caso de un proyecto de mantenimiento con las siguientes condiciones:

- $\mu_c = 100$, es decir, el cliente paga al mantenedor 150 unidades monetarias por cada h-p.
- $\mu_m = 50$, es decir, los costes del mantenedor son 50 unidades monetarias por cada h-p.
- $s = 200$, por tanto, el mantenedor sufrirá una penalización de 400 unidades monetarias por cada día de retraso.
- $n = 30$, la duración del proyecto o servicio es de 30 días.
- $h = 8$, es decir, el cliente y el mantenedor han acordado que el esfuerzo de MUC previsto se puede atender con 8 horas-persona diarias.

p	Ingresos I	Costes C(p)	Penalizaciones S(p)	Beneficios $B(p) = I - C(p) - S(p)$
1	24000	1500	651000	-628500
2	24000	3000	279000	-258000
3	24000	4500	155000	-135500
4	24000	6000	93000	-75000
5	24000	7500	55800	-39300
6	24000	9000	31000	-16000
7	24000	10500	13286	214
8	24000	12000	0	12000
9	24000	13500	0	10500
10	24000	15000	0	9000

Tabla 6-5. Resultado económico para el mantenedor con diferentes asignaciones a MUC.

Como se puede observar, el caso más favorable para el mantenedor es justo con $p=h=8$, donde obtendría un beneficio de 12000 unidades monetarias. Si se asignan más recursos de los acordados ($p>8$), no se tienen penalizaciones pero los mayores costes van reduciendo cada vez más los beneficios. Por el contrario, se observa que dedicando 1 hora-persona menos diaria el mantenedor todavía no ha entrado en pérdidas, mientras que dedicando 2 h-p diarias menos ($p=6$) las pérdidas serían de -16000 unidades monetarias. De forma exacta, el valor de p que hace que el mantenedor entre en pérdidas es $p=6,98$.

En consecuencia, esta técnica la permite saber al mantenedor que, en caso de necesidad, podría dedicar 1 hora-persona diaria menos a MUC sin llegar a la situación de pérdidas económicas propias.

El modelo propuesto realiza la simplificación de considerar que el valor de p es constante para todos los días, es decir, que el mantenedor siempre dedica las mismas h-p a atender las peticiones de MUC. La primera versión de este modelo, elaborada para el Entorno MANTIS, no realizaba esta consideración obteniéndose unas ecuaciones más complejas. Así, la ecuación (6.7) para calcular el valor umbral de p ya no era polinomial y se tenía que recurrir a métodos de optimización para resolverla. Esta primera propuesta fue modificada a la versión simplificada actual a instancias de las organizaciones que formaron el Grupo Crítico de Referencia, tal cual como se determina en el método Investigación-Acción (capítulo 2). Estas organizaciones consideraron que el anterior método complicaba en exceso los cálculos y la gestión. Además su experiencia en proyectos reales señalaba claramente que en todos los ANS reales establecidos por ellas con otras organizaciones clientes el valor de p siempre era constante.

6.2.2.1.2. Predicción del Esfuerzo de Mantenimiento en Aplicaciones Antiguas.

A continuación se presentan los resultados de un estudio empírico sobre la correlación entre métricas de código sencillas (número de módulos y líneas de código fuente) y esfuerzo de mantenimiento (Polo et al, 2001b). El objetivo de este trabajo fue proveer un método para estimar los costes de mantenimiento durante las etapas iniciales de los proyectos de mantenimiento de software con externalización, cuando el contrato de mantenimiento está siendo preparado y el mantenedor tiene poca información disponible sobre las características del software que deberá ser mantenido (el propio código fuente es muchas veces la única o casi única información).

6.2.2.1.2.1. Descripción del Experimento.

El estudio se realizó utilizando datos de proyectos reales de una de las organizaciones participantes, junto con el grupo Alarcos, en los proyectos MANTEMA, MPM y MANTIS (ver capítulo 1). Más concretamente, se trata de las aplicaciones de dos grandes bancos cuyo mantenimiento fue soportado, mediante contratos de externalización, por Atos ODS durante los años 1999 y 2000. Todos los programas estaban escritos utilizando Cobol/CICS y usando el gestor de bases de datos relacionales DB2 de IBM. Durante el tiempo de captura de datos se recibieron muchas PM's de correctivo urgente (UC), correctivo no urgente (NUC) y perfectivo (PERF), y ninguna de preventivo y adaptativo. Los datos recolectados incluyeron lo siguiente:

- número de aplicaciones de cada banco;
- número de módulos por aplicación;
- tamaño total (líneas de código fuente, LCF) de cada aplicación;
- número de PM's de los tipos UC, NUC y PERF por aplicación y mes; y
- esfuerzo total (en horas-persona) dedicado por aplicación, mes y tipo de mantenimiento.

Banco	Número de Aplicaciones	Número de Módulos	Tamaño total (LCF)
1	47	15717	10000263
2	8	1149	3832880
TOTAL	55	16866	13833143

Tabla 6-6. Principales datos del caso de estudio.

La Tabla 6-6 muestra los datos básicos del software sometido a estudio. Los datos mostrados fueron recogidos por Atos ODS mediante el cuestionario inicial (apartado 6.2.1.1.1.3). Como puede observarse, se trata realmente de aplicativos muy grandes. La Tabla 6-7 muestra el resumen de los datos recopilados sobre el mantenimiento realizado por Atos ODS en dichos proyectos. Información más detallada se puede consultar en (Polo et al, 2001b).

Banco	Nº de PM			Esfuerzo (horas-persona)			Esfuerzo medio h-p / PM		
	UC	NUC	PERF	UC	NUC	PERF	UC	NUC	PERF
1	724	1989	3884	1476	3896	47007	2.0	2.0	12.1
2	459	467	1839	1186	1209	10699	2.6	2.6	5.8
TOTAL	1183	2456	5723	2662	5105	57706	2.3	2.1	10.1

Tabla 6-7. Principales estadísticas sobre el mantenimiento realizado.

6.2.2.1.2.2. Principales Resultados.

La Tabla 6-7 muestra que el principal esfuerzo de mantenimiento no es el correctivo (en sus dos variedades) tal como a priori se podría suponer, sino el perfectivo. Esto ya ha sido manifestado con anterioridad con carácter general por diversos autores (Piattini et al, 2000; Pigoski, 1997; Basili et al, 1996).

El objetivo inicial del experimento era encontrar algún tipo de correlación entre las métricas cuya recogida es propuesta en el cuestionario inicial (variables independientes) y el número de PM's y/o el esfuerzo de mantenimiento, con el fin de estimar cuantitativamente el esfuerzo futuro previsto de mantenimiento (por ejemplo, 300 horas-persona de UC y 450 de NUC). Los resultados no permitieron establecer resultados significativos, creemos que debido a que las métricas utilizadas sólo estaban disponibles a nivel de aplicación en vez de módulo.

También se ha buscado una correlación entre el tamaño medio de los módulos de cada aplicación y el esfuerzo de mantenimiento correctivo (UC más NUC), igualmente con resultados negativos. Esto confirma que el tamaño de un módulo no es un buen predictor de su propensión a fallos, postulado ya argumentado por otros investigadores (Fenton y Ohlsson, 2000; Munson y Khoshgoftaar, 1992).

Alternativamente, se ha utilizado una técnica de *regresión logística* (Briand et al, 2000), que permite correlacionar un conjunto de variables independientes con una variable dependiente binomial. En concreto, hemos utilizado esta técnica para categorizar las aplicaciones, de acuerdo a sus necesidades previstas de mantenimiento, en dos categorías: problemática y no-problemática. Las variables independientes han sido las dos métricas disponibles: tamaño de la aplicación en LDCF, y número de módulos. Para establecer los límites entre ambas categorías se han utilizado los valores medios de PM's por mes y aplicación de nuestro caso de estudio:

$$\lambda_{UC} = 1'31 \text{ PM/mes} \quad (\text{para correctivo urgente})$$

$$\lambda_{NUC} = 4'08 \text{ PM/mes} \quad (\text{para correctivo no urgente})$$

De esta manera, si una aplicación ha originado un número mensual de PM's del tipo UC inferior a λ_{UC} se incluye en la categoría A (no-problemática). En caso contrario se incluye en la categoría B (problemática). Igual ocurre para el mantenimiento correctivo no urgente.

Con estos criterios, para el mantenimiento UC de las 55 aplicaciones consideradas, se obtuvieron los valores mostrados en la Tabla 6-8, mientras que las predicciones para el mantenimiento NUC se muestran en la Tabla 6-9.

Categoría observada	Categoría predicha		Porcentaje de aciertos
	A	B	
A	24	3	88.9%
B	7	21	75.0%
TOTAL			81.8%

Tabla 6-8. Predicciones para el mantenimiento correctivo urgente.

Categoría observada	Categoría predicha		Porcentaje de aciertos
	A	B	
A	22	5	81.5%
B	7	21	75.0%
TOTAL			78.2%

Tabla 6-9. Predicciones para el mantenimiento correctivo no urgente.

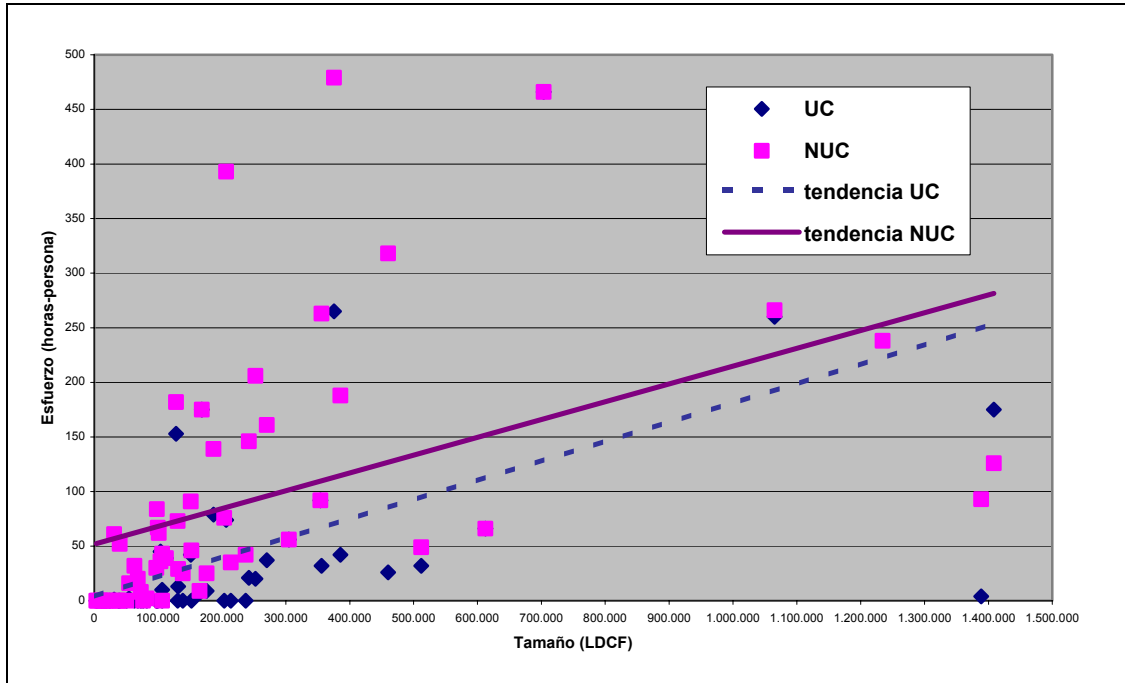


Figura 6-4. Relación entre tamaño y esfuerzo de mantenimiento correctivo.

En el mantenimiento UC, 24 de las 27 aplicaciones no problemáticas son categorizadas de forma correcta, mientras que 21 de las 28 problemática también lo son. En resumen, el porcentaje de aciertos es superior al 81%. El coeficiente de regresión es $r^2=0'67$, un valor bastante bueno. Para el mantenimiento NUC la calidad de las predicciones disminuye un poco y el coeficiente de regresión, $r^2=0'51$, ya no es tan bueno. Ambas correlaciones se muestran gráficamente en la Figura 6-4. Analizando las líneas de tendencia se concluye que el esfuerzo de mantenimiento UC crece más deprisa que el NUC al incrementar el tamaño de la aplicación.

6.2.2.1.2.3. Ecuaciones de Predicción.

Con la técnica de regresión logística es posible obtener una ecuación, como la (6.8), que devuelve uno de dos valores, 0 o 1, dependiendo de la probabilidad de que el elemento X esté clasificado en alguna de las dos categorías A o B:

$$P(x_1, x_2, \dots, x_n) = \frac{1}{1 + e^{-(w_0 + w_1 \cdot x_1 + \dots + w_n \cdot x_n)}} \quad (6.8)$$

siendo x_1, x_2, \dots, x_n las variables independientes utilizadas para la categorización y w_0, w_1, \dots, w_n los coeficientes del ajuste.

En nuestro caso de estudio, las variables independientes son el tamaño total de la aplicación (*size*) y su número de módulos (*nom*). Las ecuaciones obtenidas para mantenimiento UC y NUC son, respectivamente, las siguientes:

$$P_{UC}(\text{size}, \text{nom}) = \frac{1}{1 + e^{-(2'9822 + 0'0000387 \times \text{size} - 0'0104 \times \text{nom})}} \quad (6.9)$$

$$P_{NUC}(\text{size}, \text{nom}) = \frac{1}{1 + e^{-(2'4591 + 0'0000081 \times \text{size} - 0'0047 \times \text{nom})}} \quad (6.10)$$

6.2.3. Gestión de Riesgos.

En el ámbito de este proceso, en el Entorno MANTIS hemos incluido dos propuestas. La primera, de carácter más general, es una definición de objetivos de control para los proyectos de mantenimiento de software. La segunda, más específica, es una técnica para identificación y estimación de riesgos en proyectos con externalización del mantenimiento.

6.2.3.1. Objetivos de Control para Proyectos de Mantenimiento.

Aunque la principal función de los objetivos de control es dar soporte a los procesos de auditoría y control, también sirven para enmarcar los factores de riesgo de un proyecto. Es decir, los aspectos que se deben controlar durante la ejecución de un proyecto son precisamente aquellos que son la fuente de los principales riesgos del proyecto. Por esta razón, en el Entorno MANTIS uno de los primeros trabajos realizados consistió en la definición de los objetivos de control de los proyectos de mantenimiento de software. Estos objetivos de control se pueden utilizar para llevar a cabo la auditoría de un proyecto de mantenimiento, que no se comenta en esta tesis porque, por las razones indicadas en el capítulo 5, el proceso de auditoría queda fuera del alcance del Entorno MANTIS. También sirven para realizar el seguimiento o control del proyecto, lo que sí se enmarca dentro del alcance de MANTIS. En concreto, actividades de control se deben llevar a cabo en varios de los procesos organizacionales: gestión del proyecto, gestión de la calidad y gestión de los riesgos, (ISO/IEC, 1998c; PMI, 2000). Una presentación general de esta propuesta, tanto para la auditoría como para el seguimiento y control de proyectos de mantenimiento se puede consultar en Ruiz et al (2000a).

6.2.3.1.1. Metodología CobiT para Auditoría de Sistemas de Información.

El sistema de información de una organización es único, aunque ciertos procesos se realicen de forma manual y otros mediante la informática. Por tanto, a la hora de realizar la auditoría o control, es necesario un enfoque que considere dicho sistema de información globalmente, es decir, que tenga en cuenta, de manera conjunta, los procesos manuales y los informáticos. El auditor deberá utilizar en cada caso, las herramientas y los procedimientos más adecuados en función de la realización manual o informática de las actividades. La propuesta CobiT, “*Governance, Control and Audit for Information and related Technology*” (ISAFIC, 1998), supone un paso, seguramente el más importante, en dicho camino.

La filosofía de CobiT asimila los principios de la reingeniería de procesos de negocio (BPR, *Business Process Reengineering*), y divide las funciones que ha de realizar un sistema de información en procesos que, a su vez, están subdivididos en actividades y tareas más simples. Los sistemas de información están orientados a los procesos y por tanto su auditoría y control se debe adaptar a estos conceptos. El marco de trabajo (*framework*) de CobiT se fundamenta en una premisa simple y pragmática:

“Los recursos en Tecnologías de la Información y Comunicaciones (TIC) se han de gestionar mediante un conjunto de procesos agrupados de forma natural para que proporcionen la información que la empresa necesita para alcanzar sus objetivos”.

Para ello, se definen 34 objetivos de control generales (OCG's)²⁴, uno para cada uno de los procesos definidos. Estos procesos están agrupados en cuatro grandes dominios: planificación y organización, adquisición e implantación, suministro y soporte, y supervisión. Esta estructura cubre todos los aspectos de la información y de las tecnologías que le sirven de soporte (Peña, 1998). Además, en la estructura de CobiT se destacan los efectos de los recursos en TIC (datos, aplicaciones, tecnología, instalaciones y personal) junto con los requisitos o criterios que debe satisfacer la información:

- Requisitos de calidad: calidad, coste, suministro;
- Requisitos fiduciarios: efectividad y eficiencia de las operaciones, fiabilidad de la información, cumplimiento de las leyes y normas (COSO, 1994);
- Requisitos de seguridad: confidencialidad, integridad, disponibilidad.

En resumen, la estructura conceptual de CobiT se puede plantear desde tres puntos de vista diferentes y ortogonales (Figura 6-5):

- 1) Los recursos de las TIC,
- 2) Los criterios organizacionales de la información, y
- 3) los procesos de las TIC.

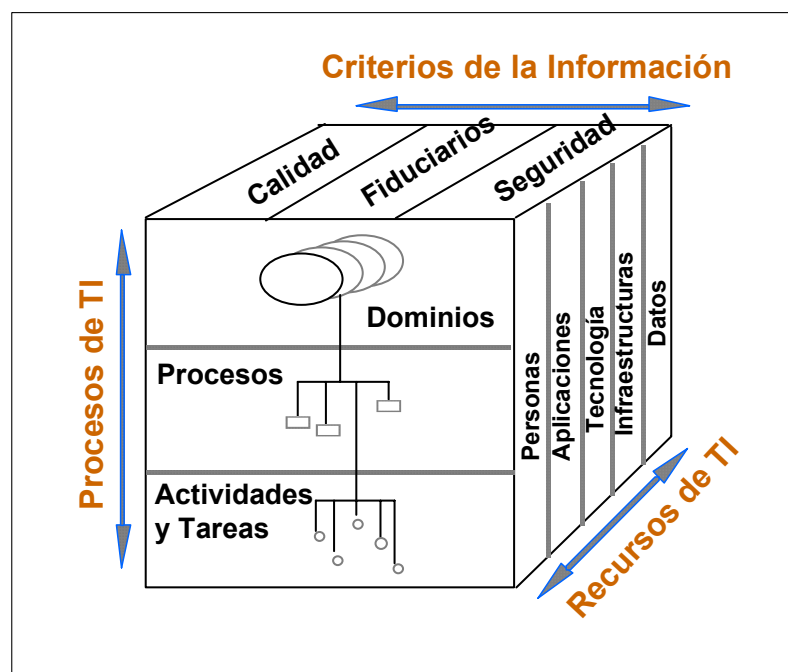


Figura 6-5. Las tres dimensiones del marco conceptual de CobiT.

²⁴ Un control se define como "las normas, estándares, procedimientos, usos y costumbres y las estructuras organizativas, diseñadas para proporcionar garantía razonable de que los objetivos empresariales se alcanzarán y que los eventos no deseados se prevenirán o se detectarán, y corregirán".

Un objetivo de control se define como "la declaración del resultado deseado o propuesto que se ha de alcanzar mediante la aplicación de procedimientos de control en cualquier actividad de las TIC".

Estas vistas diferentes permiten que se pueda utilizar el marco conceptual de manera eficiente desde la óptica de interés de los diversos implicados: directivo, gestor de TIC, responsable de procesos, técnico en TIC, o usuario.

6.2.3.1.2. Adaptación de CobiT a Proyectos de Mantenimiento.

Las principales fuentes bibliográficas dedicadas en los últimos años a la auditoría y control de sistemas de información, (Weber, 1999; Champlain, 1998; Piattini y del Peso, 1998), dedican muy poca o ninguna atención al proceso de mantenimiento del software (PMS). Ante esta situación, se optó por desarrollar una propuesta propia de adaptación de CobiT al PMS, (Ruiz et al, 1999b), que presentamos a continuación.

En CobiT, los 34 OCG's propuestos se concretan en 302 objetivos de control detallados (OCD's). En MANTIS se han utilizado, como punto de partida para la auditoría y control del PMS, los siguientes objetivos de control (seleccionados y extraídos después de una lectura y análisis minuciosos de los 34 generales y los 302 detallados):

Dominio / Objetivos Generales / Objetivos Detallados:

AI - Adquisición e Implantación

AI01 – Identificación de soluciones

1.15 Mantenimiento del software por terceros

AI02 – Adquisición y mantenimiento de aplicaciones software

2.2 Cambios grandes en sistemas existentes

AI05 – Instalación y acreditación de sistemas

5.3 Conversión

AI06 – Gestión de cambios

6.1 Iniciación y control de los requerimientos de cambio

6.2 Valorar impacto

6.3 Definir el control de cambios

6.4 Actualización de documentación y procedimientos

6.5 Autorización del mantenimiento

6.6 Política de versiones del software

6.7 Distribución del software

DS - Suministro y Soporte

DS09 – Gestión de la configuración

9.1 Registrar la configuración

9.2 Configuración básica

9.3 Contabilizar los estados pasados

9.4 Control de la configuración

9.6 Almacenar el software

Además de los anteriores, existen otros objetivos de control generales y detallados que se relacionan fundamentalmente con el proceso de desarrollo de software, pero que también son de aplicación al mantenimiento debido a que durante la actividad de realización de la modificación el mantenedor tiene que realizar algunas de las tareas típicas del desarrollo del software (análisis, diseño, codificación, prueba, etc.).

No todos los objetivos incluidos en la lista anterior tienen la misma importancia dentro del PMS (según se define en el estándar ISO 14764). El dominio en el que se incluyen la mayoría de las actividades del PMS es el de "Adquisición e Implantación". Dentro de este dominio, el OCD AI01.15 (Mantenimiento del software por terceros) pertenece realmente - a pesar del nombre- al proceso de adquisición, en este caso adquiriendo (contratando) el servicio de mantenimiento mediante externalización u *'outsourcing'*. Como ya se ha dicho, el análisis de los objetivos detallados del OCG AI02 (Adquisición y mantenimiento de aplicaciones software) permite comprobar que, también a pesar de incluir la palabra mantenimiento en el nombre, no se corresponde realmente con el PMS salvo en el OCD AI02.2 (Cambios grandes en sistemas existentes) que se refiere a situaciones que requieren mucho mantenimiento adaptativo. El OCD AI05.3 (Conversión) está relacionado con la actividad de migración dentro del PMS.

Objetivos de Control Detallados (CobiT)		Actividades relacionadas (PMS)
6.1	Iniciación y control de los requerimientos de cambio	Implementar el Proceso
6.2	Valorar impacto	Análisis del Problema y Modificación Realización de la Modificación
6.3	Definir el control de cambios	Implementar el Proceso
6.4	Actualización de documentación y procedimientos	Realización de la Modificación
6.5	Autorización del mantenimiento	Revisión/Aceptación del mantenimiento
6.6	Política de versiones del software	Implementar el Proceso
6.7	Distribución del software	Realización de la Modificación

Tabla 6-10. Objetivos de Control de la Gestión de Cambios vs Actividades del PMS.

En realidad, dentro del dominio de "Adquisición e Implantación", el OCG que realmente está asociado al PMS es el AI06 (Gestión de cambios). Todos los OCD's que lo integran están directamente asociados con las actividades del PMS. En la Tabla 6-10 se muestran los siete OCD's de la 'Gestión de cambios' junto con las actividades del PMS relacionadas. No aparece la actividad de 'migración' porque dicha actividad se produce sólo en el caso de mantenimiento adaptativo (OCD AI02.2 ya comentado). Tampoco aparece la actividad de retirada porque no se puede considerar directamente relacionada con la gestión de cambios.

En el dominio de "Suministro y Soporte", el OCG DS09 (Gestión de la configuración) está directamente relacionado con el PMS, pero se corresponde con el proceso del mismo nombre definido en la norma ISO 12207 como uno de los procesos de soporte. Por tanto, no debe ser tenido en cuenta en el proceso de mantenimiento.

6.2.3.1.3. Propuesta de Objetivos de Control.

Todas estas disfunciones se deben, fundamentalmente, a que CobiT utiliza un modelo de procesos no adaptado a los estándares de procesos de ISO y, por tanto, es muy diferente al sistema de procesos del Entorno MANTIS (capítulo 5). Por esta razón, para poder utilizar la propuesta CobiT de manera coherente con el marco conceptual de MANTIS, proponemos modificar la lista de OCG's sustituyendo el AI06 'Gestión de cambios' por 'Gestión del proceso de mantenimiento del software', en el cuál incluimos también los OCD's AI02.2 (Cambios grandes en sistemas existentes) y AI05.3 (Conversión) por las razones ya comentadas. Además, los OCD's del OCG AI06 se reestructuran en función de las actividades y tareas del PMS definidas en la norma ISO 14764 (capítulo 3).

Así, la gestión del PMS pasa a ser un objetivo de control general dentro del dominio de ‘Adquisición e Implantación’, ya que el PMS es un proceso básico para la correcta implantación (explotación) de un sistema de información, (Ruiz et al, 1999a; Ruiz et al, 2000b). A continuación se resumen los 14 objetivos de control detallados propuestos:

Dominio: Adquisición e Implantación

Objetivo General: AI06 - Gestión del proceso de mantenimiento del software

Descripción: *las actividades de la organización se realizan sin interrupciones imprevistas y el software de los sistemas de información existentes se adapta a las nuevas necesidades.*

Objetivos de Control Detallados:

6.1 Cambios en el entorno operativo: existe un procedimiento organizado para realizar la migración de un producto software desde un entorno operativo antiguo a otro nuevo.

6.2 Retirada del software: la metodología de desarrollo y/o mantenimiento de software incluye un procedimiento formal para la retirada de un producto software cuando ha concluido su ciclo de vida útil.

6.3 Tipos de mantenimiento: están categorizados los tipos de mantenimiento del software y para cada tipo se han planificado las actividades y tareas a realizar.

6.4 Contrato de mantenimiento: las relaciones entre el mantenedor y el cliente y las obligaciones de cada uno están establecidas en un contrato o acuerdo de mantenimiento.

6.5 Mejora de la calidad del proceso: la metodología empleada para el mantenimiento del software incluye técnicas para aumentar la mantenibilidad (facilidad de mantenimiento).

6.6 Planificación del mantenimiento: Existe un plan de mantenimiento que incluye el alcance del mantenimiento, quién lo realizará, una estimación de los costes y un análisis de los recursos necesarios.

6.7 Procedimientos para peticiones de modificación (PM): existen procedimientos normalizados para iniciar, recibir y registrar PM's.

6.8 Gestión y control de cambios: el mantenedor tiene establecido un interfaz organizacional para que el proceso de mantenimiento pueda verse beneficiado por el proceso de gestión de la configuración.

6.9 Análisis y valoración de las PM's: las PM's son categorizadas y priorizadas, y existen mecanismos bien estructurados para evaluar su impacto, costes y criticidad.

6.10 Verificación de los problemas: el mantenedor replica o verifica que realmente existe el problema que originó la PM.

6.11 Registro de las PM's: el mantenedor documenta y registra las PM's, con sus análisis, valoraciones y verificaciones.

6.12 Aprobación: dependiendo del tipo de mantenimiento de una PM, existen procedimientos formales que detallan el tipo de aprobación que el mantenedor debe obtener antes y después de realizar la modificación.

6.13 Realización de las modificaciones: para realizar las modificaciones, el mantenedor utiliza la misma metodología establecida para el proceso de desarrollo del software adaptada al proceso de mantenimiento.

6.14 Actualización de la documentación: la documentación (informes técnicos, manuales, etc.) afectada por una PM es actualizada después de realizada la modificación.

En conclusión, podemos decir que el objetivo de control general propuesto busca que la gestión del PMS permita que las actividades de la organización se realicen sin interrupciones imprevistas y que el software de los sistemas de información existentes se adapte a las nuevas necesidades; teniendo en consideración los cambios en el entorno operativo, cómo se hace la retirada del software, los tipos de mantenimiento, etc.

Para cualquier organización que tenga responsabilidades de mantenimiento de algún producto software, ya sean internas (un departamento de la propia organización es el responsable del mantenimiento) o externas (el servicio se ofrece a otra organización diferente), los objetivos de control propuestos se pueden utilizar al estilo de una lista de comprobación o ‘check-list’. Por ejemplo, si se utiliza la metodología general de Auditoría de Sistemas de Información propuesta por Weber (1999), los 14 objetivos de control anteriores serán útiles en las tareas siguientes:

- Estimación de la calidad de los controles implementados sobre el PMS; y
- Comprobación de que los controles sobre el PMS son realmente cumplidos en todos los sitios donde se realiza el mantenimiento.

Por otro lado, los responsables de la auditoría y control deberán ser cuidadosos de no centrarse exclusivamente en la seguridad e integridad de datos (los dos aspectos más frecuentes contemplados), ya que en el caso del PMS, la gran cantidad de recursos consumidos habitualmente en las actividades de mantenimiento, obliga a dichos responsables a considerar también la efectividad y eficiencia con que cada actividad de mantenimiento es realizada por cada participante.

6.2.3.2. Riesgos en la Externalización de Servicios de Mantenimiento.

En el capítulo 4 se han comentado algunos buenos trabajos referidos a los riesgos una vez el proyecto de mantenimiento ha comenzado. Ejemplos significativos de ello son las propuestas de Schneidewind (1997) y Sherer (1997). Pero existe una carencia de guías para ayudar a los gestores a identificar y estimar los riesgos durante las etapas iniciales de los proyectos de mantenimiento. Esto es un problema especialmente importante para las organizaciones que realizan servicios de mantenimiento mediante externalización cuando tienen que realizar las ofertas de mantenimiento y determinar los acuerdos de nivel de servicio. Quizás la única excepción a dicha situación sean los trabajos de Briand et al (1998a) y Shepherd et al (1996), si bien ambos proponen identificar y estimar los riesgos mediante un “juicio de experto”, sin aportar muchos más detalles.

El proceso de gestión de riesgos se suele considerar formado por varias actividades. Por ejemplo, PMBOK (PMI, 2000) considera las cuatro siguientes:

1. identificación de los riesgos;
2. estimación o cuantificación de los riesgos;
3. desarrollo de respuestas; y
4. control de las respuestas.

Puesto que el objetivo del procedimiento propuesto es ayudar a elaborar el contrato de mantenimiento, sólo se consideran las dos primeras actividades (Polo et al, 2002b). En el

momento de realizar dichas actividades no se tiene buena información para hacer una estimación razonable de la exposición a cada riesgo, es decir, de la probabilidad de que ocurra multiplicada por su impacto en unidades monetarias o de costes de recursos. Por esta razón, para la segunda actividad, en lugar de una cuantificación, se propone un procedimiento de priorización, que permite determinar los riesgos principales para el mantenedor.

6.2.3.2.1. Identificación de los Riesgos.

La idea básica es proponer al mantenedor una colección de factores de riesgo que pueda utilizar como lista de control (*check-list*). En *Euromethod*²⁵ (EC, 1996) se señala que los factores situacionales son “aquellas propiedades de la situación que generan riesgos y que deberán ser tenidas en cuenta en el diseño de la estrategia (léase ‘planificación del proyecto’)”. Por tanto, un factor situacional puede afectar a uno o varios factores de riesgo que, a su vez, pueden afectar a una o más áreas de negocio (Neitzel, 1999). La conclusión es que es importante determinar y comprender las relaciones entre los factores situacionales y los riesgos del proyecto.

La primera versión de la lista de control incluía 36 preguntas elaboradas con algunos de los factores situacionales de *Euromethod* y con otros factores propuestos por Atos ODS. Estos últimos se agruparon, mediante una técnica de análisis de componentes principales (Sierra, 1994), en categorías de factores que miden la misma propiedad del proyecto. Esa primera versión fue entregada a los responsables de 16 grandes proyectos. 12 de los proyectos, pertenecientes a Atos ODS, estaban desarrollados íntegramente en Cobol, Cobol-CICS y Cobol-CICS-DB2, mientras que los otros 4, pertenecientes a la Diputación Provincial de Ciudad Real, estaban desarrollados en un entorno 4GL. Cada uno de los proyectos correspondía al mantenimiento de un producto software con un tamaño comprendido entre 125000 y 275000 LDCF.

Un nuevo análisis de componentes principales se llevó a cabo con los datos recolectados de los 16 proyectos, obteniéndose seis componentes principales que acumulaban más del 98% de la varianza total. Como resultado del análisis, se obtuvo un cuestionario que reducía los 36 factores situacionales iniciales a sólo 21 esenciales.

Grupo		Factor situacional	Valor	Media
SPEC	Especificaciones existentes de la aplicación	El diseño de la arquitectura técnica del sistema es de alta calidad		
		El diseño del software es de alta calidad		
		Los requisitos del sistema están disponibles y son claros		
		Los requisitos del sistema son estables		
TEAM	Equipo actual de mantenimiento y/o desarrollo del cliente	El equipo tiene una alta capacidad de trabajo		
		El equipo tiene un conocimiento elevado del dominio del sistema		
		El equipo tiene un conocimiento elevado del entorno tecnológico		
EXT	Entorno exterior	La organización depende poco de subcontratistas		

²⁵ En la actualidad, *Euromethod* ha sido sustituido por el proyecto ISPL (*Information Services Procurement Library*), también de la Comisión Europea.

Grupo		Factor situacional	Valor	Media
IS	Sistema de Información	Los procesos de negocio son sencillos		
		La información es estable		
CRI	Criticidad de la aplicación	No existen momentos críticos aleatorios		
		No existen picos periódicos que requieran la incorporación de más personal o la dedicación de mayores recursos (por ejemplo, a finales de mes)		
		No se depende de los usuarios para conocer el dominio del problema		
QUA	Calidad de la aplicación	Los estándares de codificación y programación son ampliamente usados		
		La calidad de los programas “batch” es alta		
		Las pantallas y formularios son sencillos y están poco afectados por intervenciones de mantenimiento		
ORG	La organización	Existe poca dependencia respecto a los cambios en la organización		
		Los procesos de negocio tienen pocos cambios		
MET	Metodologías	La tecnología necesaria en el proyecto está disponible		
		Durante el desarrollo del sistema se utilizaron metodologías contrastadas		
		Durante el desarrollo del sistema se utilizaron herramientas adecuadas		

Tabla 6-11. Cuestionario con los factores situacionales para externalización del mantenimiento.

El nuevo cuestionario resultante es mostrado en la Tabla 6-11. La primera columna muestra los grupos de factores identificados, que se desglosan en uno o varios factores situacionales mostrados en la columna segunda. Cada celda en la columna tercera debe ser completada con un valor entre 1 y 5, según el encuestado esté completamente de acuerdo, de acuerdo, le sea indiferente, en desacuerdo, o completamente en desacuerdo, con la afirmación indicada por el factor situacional. La última columna se completa mediante el cálculo de una media aritmética o ponderada de las respuestas anteriores. Los pesos utilizados para calcular la media son decididos por cada organización a partir de su experiencia en proyectos anteriores.

6.2.3.2.2. Priorización de los Riesgos.

Una vez se ha rellenado la Tabla 6-11, la siguiente etapa es la priorización de los riesgos. Para ello se tiene en cuenta que los grupos de factores con valores medios más bajos son los que suponen mayores debilidades de la organización cliente y, por tanto, mayores peligros (riesgos) para la organización mantenedor.

Cada grupo de factores influye en la posibilidad de que se materialicen diversos tipos de riesgos. Por esta razón, en la Tabla 6-12 se refleja la correspondencia entre los grupos de factores y los riesgos más frecuentes en proyectos de mantenimiento. Esta correspondencia se ha obtenido mediante un análisis estadístico de los datos de los 16 proyectos citados anteriormente.

Cada riesgo es priorizado copiando los valores de la columna cuarta del cuestionario anterior correspondientes a los grupos de factores que tienen su casilla en blanco en la Tabla 6-12 para dicho riesgo. A continuación, en la columna “suma total” se hace la suma de los

valores copiados y en la última columna se muestra el resultado de dividir por el número indicado al lado. Este número viene dado por la cantidad de celdas en blanco multiplicadas por 5 (que era el valor máximo en el cuestionario primero). El valor final obtenido, es decir, el valor del riesgo, es un indicador de la prioridad (medida en tanto por uno) de dicho riesgo. Cuanto más cercano a 1 sea mejor serán los resultados.

		Grupos de factores situacionales							Suma Total	Valor del Riesgo	
Riesgos		SPEC	TEAM	EXT	IS	CRI	QUA	ORG	MET		
R1	Requisitos ambiguos o irrealizables									/10	
R2	Interfaces ambiguos con otros sistemas									/5	
R3	Requisitos cambiantes									/20	
R4	Costes impredecibles para el mantenedor									/20	
R5	Costes impredecibles para el proyecto									/20	
R6	Retrasos en el servicio									/15	
R7	Baja calidad de los entregables									/15	
R8	Costes crecientes en el proyecto									/10	
R9	Problemas de integración									/10	
R10	Capacidades tecnológicas llevadas al límite									/5	
R11	El S.I. erróneo o irrealizable									/20	
R12	Los actores no aceptan adaptarse al nuevo S.I.									/5	
R13	El proyecto fracasa por implicaciones de negocio									/10	

Tabla 6-12. Cuantificación de los riesgos de la externalización a partir de los factores situacionales.

Por ejemplo, si los valores medios obtenidos con la Tabla 6-11 son $EXT=3'5$, $IS=2'6$, $CRI=2'1$, $ORG=1'8$, etc., entonces la priorización de los riesgos sería

$$R1(\text{requisitos ambiguos o irrealizables}) = (3'5+2'6)/10 = 0'61$$

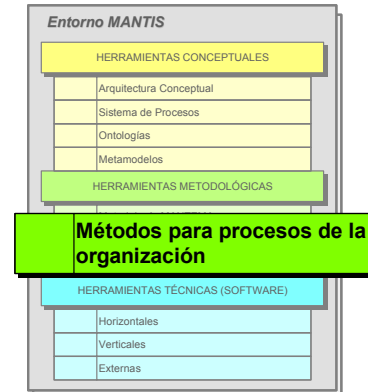
$$R3(\text{requisitos cambiantes}) = (3'5+2'6+2'1+1'8)/20 = 0'50$$

Este resultado indica que el mantenedor debería priorizar el riesgo R3 por delante del R1 ya que la situación prevista es potencialmente más peligrosa para R3 que para R1.

En esta técnica no se realiza una cuantificación de los riesgos indicando una estimación de sus efectos en caso de ocurrir (normalmente medidos en retrasos en horas-persona). Se deja a la elección de cada organización mantenedor la decisión de si con la priorización les es suficiente o prefieren completarla con la cuantificación. En este segundo caso se pueden utilizar técnicas de correlación para, basándose en los resultados de proyectos históricos, comparar los valores de los riesgos obtenidos en la Tabla 6-12 con los retrasos reales ocurridos, y obtener los retrasos estimados para cada riesgo.

6.3. Métodos para los Procesos de la Organización.

En esta parte se presentan los procedimientos definidos para los procesos de mejora, gestión de recursos humanos y medición. Dentro del alcance de la versión actual del Entorno MANTIS no se han creado nuevos procedimientos utilizables dentro de los procesos de alineamiento organizacional, infraestructura y reutilización. Ahora bien, en cierto sentido, los procedimientos de uso de las herramientas presentadas en el capítulo 7 se pueden considerar dentro del proceso de infraestructura.



6.3.1. Mejora.

En la versión actual del Entorno MANTIS, la mejora del PMS se aborda desde las dos perspectivas de mejora de servicios propuestas por Niessink (2000):

- a) La mejora basada en la medida utiliza el modelo presentado en el capítulo 4. De esta manera, son los objetivos de mejora los que determinan las métricas a utilizar (y las observaciones para obtener los valores). A su vez, los resultados de las mediciones son necesarios para poder analizar e implementar las mejoras propuestas. Esta es la razón por la que el marco conceptual de MANTIS otorga una importancia especial al proceso de medición. Prueba de ello es la incorporación de la ontología de la medida presentada en el capítulo 5, incluyendo la conceptualización del proceso de medición también presentada en dicho capítulo. Esta última está adaptada al modelo de Niessink de mejora basada en la medida. En el apartado 6.3.3, dedicado al proceso de medida, se detallan más estos aspectos.
- b) La mejora basada en la madurez también utiliza las propuestas de Niessink (2000). En concreto, el modelo de madurez de servicios de tecnologías de la información ITS-CMM (*IT Service Capability Maturity Model*) es una de las bases de trabajo, junto con los estándares CMM (Paulk et al, 1993) e ISO 15504 (ISO/IEC, 1998b) y algunas otras propuestas más concretas, de los procedimientos y casos de estudio que se presentan a continuación, en este mismo apartado.

Es conocido que una de las maneras de mejora de procesos es que dichos procesos sean repetibles (nivel 2 de CMM) y estén bien definidos (nivel 3 de CMM). Por esta razón, una primera aportación del Entorno MANTIS a la mejora del PMS es la propia metodología MANTEMA, que establece un modelo de proceso de mantenimiento detallado y preciso. En (Polo et al, 2000c) se pueden consultar las ventajas, a juicio de los autores, de utilizar MANTEMA según los modelos de madurez CMM y Bootstrap (Zahran, 1997), es decir, cómo el uso de MANTEMA ayuda a una organización mantenedor a pasar a los niveles 2 y 3.

La segunda aportación genérica del Entorno MANTIS a la mejora del PMS es consecuencia de incluir las ontologías presentadas en el capítulo anterior. Tal como afirman Bertrand y Bézivin (2000), las ontologías son una herramienta clave para mejorar la comunicación entre los diversos actores (humanos o automáticos) que pueden participar en un proyecto, es decir, son una ayuda en uno de los aspectos clave de la gestión de cualquier tipo de

proyectos. Esta razón se ha justificado más en detalle en el apartado del capítulo 5 que explica el uso de las ontologías en MANTIS.

6.3.1.1. Guía para Evaluar la Madurez de un Servicio de Mantenimiento.

A continuación, se presenta un procedimiento para evaluar la madurez del servicio de mantenimiento proporcionado por una organización mantenedor. El procedimiento se basa en la utilización de un cuestionario, que ha sido desarrollado utilizando el modelo de madurez ITS-CMM de Niessink y Van Vliet (1999b; 2000). Así, este cuestionario se propone en el Entorno MANTIS como una guía para el proceso de mejora. Con las respuestas al cuestionario, y siguiendo las directrices propuestas por ITS-CMM, es posible evaluar la adecuación a los niveles 2 y/o 3 de un servicio de mantenimiento de software.

En concreto, el cuestionario se ha elaborado a partir de las 12 áreas de proceso claves (KPA's, "Key Process Areas") que los citados autores proponen para los niveles 2 y 3. Estos niveles de ITS-CMM se corresponden con los mismos de CMM, pero adaptados para el mantenimiento en el sentido de que se considera que se trata más de un servicio que de un proceso de producción, como sería el caso del desarrollo de software (ver capítulo 4 al respecto). En la Tabla 6-13 se muestran de forma resumida dichas KPA's, y la cantidad de objetivos, compromisos, aptitudes, actividades, medidas y verificaciones propuestas para cada KPA.

Nivel	Área de Proceso Clave (KPA)	Objetivos	Compromisos	Aptitudes	Actividades	Medidas	Verificaciones
2	Gestión de compromisos de servicio	2	2	3	5	1	3
	Planificación de la entrega de servicios	3	2	3	10	1	3
	Servicio de seguimiento y detección de errores	3	2	5	14	1	3
	Gestión de subcontratos	4	2	3	12	2	3
	Gestión de la configuración	4	1	4	10	1	4
	Gestión de eventos	3	1	3	7	2	4
	Aseguramiento de la calidad en el servicio	4	2	4	7	1	3
	TOTAL NIVEL 2	23	12	25	65	9	23
3	Definición del proceso organizacional	2	1	2	6	1	1
	Enfoque en el proceso organizacional	3	3	4	7	1	2
	Programa de entrenamiento	3	1	4	6	2	2
	Gestión integrada de los servicios	2	1	3	11	1	3
	Entrega de servicios	4	2	3	10	1	3
	TOTAL NIVEL 3	14	8	16	40	6	11
TOTAL		37	20	41	105	15	34

Tabla 6-13. Áreas de proceso clave para los niveles 2 y 3 de ITS-CMM.

6.3.1.1.1. Preguntas del Cuestionario.

El cuestionario tiene 105 preguntas, 65 de las cuales corresponden al nivel 2 de ITS-CMM y 40 al nivel 3. Cada pregunta tiene una respuesta a elegir entre las 4 siguientes:

- *Sí siempre*, si la actividad es realizada siempre. En este caso además se pide indicar el responsable.

- *No siempre*, si la actividad se lleva a cabo algunas veces o no se realiza de forma completa. En este caso se pide indicar las causas de que dicha actividad no se realice de forma consistente.
- *Nunca*, si la actividad no es realiza nunca o prácticamente nunca.
- *No sabe*, si no se está seguro de lo que significa la actividad o no se tiene suficiente información para responder.

Con las respuestas al cuestionario se procede a realizar la evaluación de la madurez respecto de los niveles 2 y 3. En MANTIS se propone utilizar exactamente el mismo procedimiento que utiliza CMM, que también es el sugerido por Niessink y Van Vliet (2000).

El cuestionario atiende a varios aspectos de gestión: gestión de acuerdos de nivel de servicio (ANS), planificación de entregas de servicio, supervisión del servicio, gestión de subcontratos, gestión de eventos (en nuestro caso, peticiones de modificación) y aseguramiento de la calidad de servicio. También atiende al proceso de soporte de gestión de la configuración, dada su especial relevancia para el mantenimiento. Las preguntas, agrupadas por nivel y KPA, son las siguientes:

6.3.1.1.1. Preguntas del Nivel 2.

2.1 KPA: Gestión de compromisos de servicio:

- 2.1.1 ¿Existe un procedimiento documentado para identificar los servicios IT que el cliente necesita?
- 2.1.2 ¿Están documentadas las necesidades de servicios IT del cliente?
- 2.1.3 ¿Están documentados los compromisos de servicio?
- 2.1.4 ¿Son evaluados, de manera conjunta con el cliente, los compromisos de servicio, cada cierto periodo de tiempo?
- 2.1.5 ¿Se evalúa, de manera conjunta con el cliente, la entrega real de servicios, cada cierto periodo de tiempo?

2.2 KPA: Planificación de la entrega de servicios:

- 2.2.1 ¿Existe un procedimiento documentado para desarrollar el plan de entrega de servicios?
- 2.2.2 ¿Está documentado el plan de entrega de servicios?
- 2.2.3 ¿Las actividades de entrega de servicios están siendo identificadas, planeadas y ejecutadas según un procedimiento documentado?
- 2.2.4 ¿Están disponibles los productos hardware y software necesarios para establecer y mantener un control de la entrega de servicios?
- 2.2.5 ¿Son estimadas las cargas de trabajo derivadas de la entrega de servicios según un procedimiento documentado?
- 2.2.6 ¿Son estimados el esfuerzo y los costes derivados de la entrega de servicios según un procedimiento documentado?
- 2.2.7 ¿Se planifica la agenda de la entrega de servicios según un procedimiento documentado?
- 2.2.8 ¿Están identificados, evaluados y documentados los riesgos asociados a los costes, recursos, tiempos y aspectos técnicos de los servicios?
- 2.2.9 ¿Están preparados los planes y las herramientas de soporte para facilitar los servicios?

2.2.10 ¿Están protegidos (resguardados) los datos de planificación de los servicios?

2.3 KPA: Servicio de seguimiento y detección de errores:

- 2.3.1 ¿Existe un plan documentado de entrega de servicios y es utilizado para supervisar las actividades de entrega de servicios y notificar su estado?
- 2.3.2 ¿Es revisado el plan de entrega de servicios según un procedimiento documentado?
- 2.3.3 ¿Se comunican los cambios aprobados en la entrega de servicios a los miembros del grupo de entrega de servicios y a otros grupos relacionados?
- 2.3.4 ¿Están supervisados los niveles de servicio según las especificaciones correspondientes y se toman acciones correctivas en caso necesario?
- 2.3.5 ¿Están supervisadas las cargas de trabajo de la entrega de servicios y se toman acciones correctivas en caso necesario?
- 2.3.6 ¿Están supervisados los costes y esfuerzos de las actividades de entrega de servicios y se toman acciones correctivas en caso necesario?
- 2.3.7 ¿Están supervisados los recursos necesarios para desarrollar los servicios y se toman acciones correctivas en caso necesario?
- 2.3.8 ¿La agenda de entrega de servicios es supervisada y se toman acciones correctivas cuando es necesario?
- 2.3.9 ¿Las actividades de entrega de servicios son supervisadas y se toman acciones correctivas cuando es necesario?
- 2.3.10 ¿Se supervisan los riesgos asociados a los costes, recursos, tiempos y aspectos técnicos de los servicios?
- 2.3.11 ¿Se registran y están disponibles en caso necesario las mediciones de los datos de servicio y su re-planificación?
- 2.3.12 ¿Es sometido periódicamente a revisiones internas el grupo de entrega de servicios para supervisar el estado de las actividades, planes, niveles de servicio actuales, y comprobar que se adaptan al plan de entrega de servicios?
- 2.3.13 ¿Se realizan, según un procedimiento documentado, revisiones formales conjuntas con el cliente para evaluar los resultados y grado de cumplimiento de la entrega de servicios?
- 2.3.14 ¿Se realizan, cada cierto periodo de tiempo, revisiones formales internas siguiendo un procedimiento documentado?

2.4 KPA: Gestión de subcontratos:

- 2.4.1 ¿Se especifica y planifica el servicio subcontratado según un procedimiento documentado?
- 2.4.2 ¿Se selecciona al subcontratado para la entrega de un servicio según un procedimiento documentado y según las habilidades del mismo respecto a otros candidatos?
- 2.4.3 ¿Es el acuerdo o contrato con el subcontratado la base para la gestión del servicio correspondiente?
- 2.4.4 ¿Está revisado y aprobado el plan de entrega de servicios del servicio subcontratado?
- 2.4.5 ¿Se utiliza el documento que contiene el plan de entrega de servicios del servicio subcontratado para supervisar las actividades del servicio y comunicar su estado?
- 2.4.6 ¿Son resueltos, según un procedimiento documentado, los cambios en los compromisos del servicio, en el plan de entrega del servicio o en otros compromisos?
- 2.4.7 ¿Son evaluados, conjuntamente con el subcontratado, los compromisos del servicio subcontratado cada cierto periodo de tiempo?

- 2.4.8 ¿Es evaluada la entrega de servicios, conjuntamente con el subcontratado, cada cierto periodo de tiempo?
- 2.4.9 ¿Se gestionan formalmente, según un procedimiento documentado, las revisiones del cumplimiento de los objetivos y los resultados obtenidos en los servicios subcontratados?
- 2.4.10 ¿El grupo de aseguramiento de la calidad, asegura la calidad de las actividades del servicio subcontratado siguiendo un procedimiento documentado?
- 2.4.11 ¿El grupo de gestión de la configuración, gestiona las actividades de configuración del servicio subcontratado según un procedimiento documentado?
- 2.4.12 ¿El grupo de gestión de eventos, gestiona los eventos de las actividades del servicio subcontratado según un procedimiento documentado?.

2.5 KPA: Gestión de la configuración:

- 2.5.1 ¿Se prepara, siguiendo un procedimiento documentado, un plan de gestión de la configuración para cada servicio?
- 2.5.2 ¿Existen un documento y un plan de gestión de la configuración aprobados y se utilizan como bases para el desarrollo de las actividades de gestión de la configuración?
- 2.5.3 ¿Existe una biblioteca permanente del sistema de gestión de la configuración que hace de repositorio para las líneas base de configuración?
- 2.5.4 ¿Están identificados los productos que se obtienen como resultado de la gestión de la configuración?
- 2.5.5 ¿Son almacenados, revisados, aprobados y supervisados los elementos software de la gestión de la configuración siguiendo un procedimiento documentado?
- 2.5.6 ¿Se controlan los cambios en la gestión de la configuración de acuerdo con un procedimiento documentado?
- 2.5.7 ¿Son creados y desarrollados los productos software sometidos a gestión de la configuración de acuerdo con un procedimiento documentado?
- 2.5.8 ¿Se registra el estado de los elementos o items de configuración registrado según un procedimiento documentado?
- 2.5.9 ¿Están estandarizados los documentos de gestión de la configuración y las líneas base de la configuración desarrolladas de acuerdo con los grupos y los individuos afectados?
- 2.5.10 ¿Están las líneas base de la configuración auditadas y gestionadas según un procedimiento documentado?

2.6 KPA: Gestión de eventos:

- 2.6.1 ¿Existe un plan de gestión de eventos preparado para cada servicio de acuerdo con un procedimiento documentado?
- 2.6.2 ¿Están desarrollados el documento y el plan de gestión de eventos según las bases para la gestión de eventos?
- 2.6.3 ¿Se graban los eventos ocurridos en una biblioteca que es utilizada como repositorio para la gestión de eventos?
- 2.6.4 ¿Son identificados, registrados, analizados, revisados y resueltos los eventos de acuerdo con un procedimiento documentado?
- 2.6.5 ¿Son los grupos e individuos interesados informados del estado de los eventos cada cierto periodo de tiempo?
- 2.6.6 ¿Existen documentos e informes estandarizados de las actividades de gestión de eventos y de los contenidos del repositorio, y están disponibles para los grupos e individuos afectados?

- 2.6.7 ¿Es auditado y gestionado el repositorio de eventos de acuerdo a un procedimiento documentado?

2.7 KPA: Aseguramiento de la calidad en el servicio:

- 2.7.1 ¿Está el plan de aseguramiento de la calidad del software (SQA) preparado para la entrega de servicios de acuerdo con un procedimiento documentado?
- 2.7.2 ¿Están desarrolladas las actividades del grupo SQA de acuerdo con el plan de SQA?
- 2.7.3 ¿Participa el grupo SQA en la preparación y revisión de los compromisos y en la planificación de la entrega de servicios, estándares y procedimientos?
- 2.7.4 ¿Audita el grupo SQA las actividades de entrega de servicios para verificar su satisfacción?
- 2.7.5 ¿Comunica periódicamente el grupo SQA los resultados de sus actividades a los grupos de entrega de servicios?
- 2.7.6 ¿Están documentadas las desviaciones identificadas en las actividades de los servicios y en los servicios de entrega, y son gestionadas según un procedimiento documentado?
- 2.7.7 ¿Realiza el grupo SQA revisiones periódicas de sus actividades junto con el cliente?

6.3.1.1.1.2. Preguntas del Nivel 3.

3.1 KPA: Definición del proceso organizacional:

- 3.1.1 ¿Se desarrolla y mantiene, según un procedimiento documentado, un proceso software organizacional estándar?
- 3.1.2 ¿Está documentado el proceso software organizacional estándar según los estándares establecidos por la organización?
- 3.1.3 ¿Se documentan y se mantienen las descripciones de los ciclos de vida que se siguen en cada uno de los proyectos?
- 3.1.4 ¿Se desarrollan y se mantienen guías de consulta y criterios para la adaptación del proceso software organizacional estándar a los procesos de cada proyecto?
- 3.1.5 ¿Existe y se mantiene una base de datos del proceso software organizacional estándar?
- 3.1.6 ¿Existe y se mantiene una biblioteca con la documentación relacionada con el proceso software?

3.2 KPA: Enfoque en el proceso organizacional:

- 3.2.1 ¿Se evalúa periódicamente el proceso software y se desarrollan planes de acción para tratar los imprevistos?
- 3.2.2 ¿Existe y se mantiene un plan para las actividades de desarrollo y mejora del proceso software?
- 3.2.3 ¿Se coordinan a nivel organizacional las actividades y los proyectos para el desarrollo y mejora de los procesos software?
- 3.2.4 ¿Se coordina a nivel de la organización el uso de la base de datos del proceso software organizacional?
- 3.2.5 ¿Se vigila, evalúa y limita el uso de procesos, métodos y herramientas nuevos, antes de transferirlos a otras partes de la organización?
- 3.2.6 ¿Se coordina a nivel organizacional el entrenamiento para los procesos software y para los proyectos?
- 3.2.7 ¿Los grupos involucrados en la implementación del proceso software son informados de las actividades y proyectos de la organización para el desarrollo y mejora de software?

3.3 KPA: Programa de entrenamiento:

- 3.3.1 ¿Existe y se mantiene un plan de entrenamiento para cada proyecto software que cubre sus necesidades de entrenamiento específicas?
- 3.3.2 ¿Se desarrolla y se revisa, según un procedimiento documentado, el plan de entrenamiento de la organización?
- 3.3.3 ¿Se desarrolla el entrenamiento en la organización de acuerdo con el plan del programa de entrenamiento?
- 3.3.4 ¿Se preparan los cursos del programa de entrenamiento según el nivel organizacional y se desarrollan y se mantienen según los estándares de la organización?
- 3.3.5 ¿Se establecen y se usan los procedimientos requeridos en el programa de entrenamiento para determinar los conocimientos que poseen los individuos y las habilidades que deben adquirir para desempeñar sus roles?
- 3.3.6 ¿Se registran y se mantienen los datos relativos al programa de entrenamiento?

3.4 KPA: Gestión integrada de los servicios:

- 3.4.1 ¿Se adapta el proceso software de la organización al proceso de servicios IT según un procedimiento documentado?
- 3.4.2 ¿Se revisan, según un procedimiento documentado, las actividades del proceso de servicios IT?
- 3.4.3 ¿Se desarrolla y revisa, según un procedimiento documentado, un plan de trabajo para el proceso de servicios IT?
- 3.4.4 ¿Se desarrolla el proceso de servicios IT de acuerdo con el proceso software del proyecto?
- 3.4.5 ¿Se utiliza la base de datos organizacional para planificar y estimar el proceso de servicios IT?
- 3.4.6 ¿Se gestiona el alcance de los servicios IT según un procedimiento documentado?
- 3.4.7 ¿Se gestionan el esfuerzo y los costes del proceso de servicios IT según un procedimiento documentado?
- 3.4.8 ¿Se gestionan los imprevistos en el proceso de servicios IT de acuerdo con un procedimiento documentado?
- 3.4.9 ¿Se gestionan las dependencias temporales y las rutas críticas en el plan (*scheduling*) del proceso de servicios IT según un procedimiento documentado?
- 3.4.10 ¿Se identifican, evalúan, documentan y gestionan los riesgos del proceso de servicios IT según un procedimiento documentado?
- 3.4.11 ¿Se efectúan revisiones periódicas para determinar acciones que desarrollen el proyecto según lo planeado?

3.5 KPA: Entrega de servicios:

- 3.5.1 ¿Se lleva a cabo la entrega de servicios de acuerdo con un procedimiento documentado?
- 3.5.2 ¿Las actividades de la entrega de servicios se identifican y planifican según un procedimiento documentado?
- 3.5.3 ¿Se dispone de los recursos hardware y software necesarios para la entrega de servicios?
- 3.5.4 ¿Se realiza la correspondiente documentación de la entrega de servicios de acuerdo a un procedimiento documentado?
- 3.5.5 ¿Se intentan cumplir las estimaciones de carga de trabajo para la entrega de servicios o, en su defecto, se adaptan a los cambios según un procedimiento documentado?
- 3.5.6 ¿Se intentan cumplir las estimaciones de esfuerzo para la entrega de servicios o, en su defecto, se adaptan a los cambios según un procedimiento documentado?

- 3.5.7 ¿Se procura respetar la agenda para la entrega de servicios o, en su defecto, se adapta a los cambios según un procedimiento documentado?
- 3.5.8 ¿Están identificados, evaluados y documentados los riesgos asociados a los costes, recursos, agenda y aspectos técnicos?
- 3.5.9 ¿Se aplican los planes y herramientas de soporte si ocurren imprevistos?
- 3.5.10 ¿Se registran todos los datos relativos a la entrega de servicios?

6.3.1.1.2. Caso de Aplicación.

El cuestionario presentado se utilizó para evaluar la madurez del servicio de mantenimiento del software de 2 organizaciones públicas: el Centro Provincial de Informática (CENPRI), perteneciente a la Diputación Provincial de Ciudad Real (uno de los miembros del Grupo Crítico de Referencia de los proyectos del Entorno MANTIS), y un segundo, que prefiere permanecer en el anonimato.

El CENPRI mantiene software, desarrollado por ellos mismos o adquirido a terceros, para la gestión del gobierno provincial, los 102 municipios de la provincia y las relaciones con los ciudadanos. Esto incluye recaudación de impuestos, embargos de propiedades, infraestructuras y servicios públicos, permisos para obras privadas, salarios y gestión del personal. También se refiere al software para interaccionar con otras organizaciones: bancos, gobierno regional, ministerios, etc.

El segundo centro desarrolla y mantiene software para la gestión de una institución pública bastante grande, con varios miles de empleados y con muchos tipos diferentes de relaciones con los ciudadanos y con otras instituciones.

Ambas organizaciones utilizan herramientas para registrar y supervisar los informes de problemas recibidos de los usuarios. El centro anónimo utiliza una aplicación web para introducirlos, mientras que el CENPRI recibe dichos informes por teléfono, fax o email. Las peticiones de modificación (PM's) recibidas por ambos centros, durante el periodo de estudio, estuvieron repartidas por tipos de mantenimiento según se muestra en la Tabla 6-14.

Tipo de Mantenimiento	Peticiones de Modificación (%)
Correctivo	13.3
Perfectivo	62.2
Adaptativo	13.2
Preventivo	11.3

Tabla 6-14. Distribución de las PM's por tipos de mantenimiento.

En Polo et al (2001c) se pueden consultar los resultados detallados obtenidos de aplicar el cuestionario a las dos organizaciones citadas. Las principales conclusiones, útiles para otros casos similares, fueron las siguientes:

- Los departamentos de IT de instituciones públicas tienen características especiales que podrían requerir adaptaciones más específicas de los modelos de madurez existentes. Por ejemplo, el software que producen y mantienen es de un tipo concreto y bastante parecido entre sí.

- b) En ambos casos, al ser la misma organización el mantenedor y el cliente, existe una proximidad entre el equipo de mantenimiento y los usuarios que hace que compartan objetivos y los problemas de comunicación sean menores que otros casos.
- c) Estos departamentos cumplen especialmente bien la documentación de aquellas KPA's sujetas a obligaciones legales como, por ejemplo, la gestión de subcontratos.
- d) Estos departamentos suelen tener muy poca movilidad del personal, lo cual genera la "costumbre" de documentar menos todavía que en otras organizaciones, confiando que las personas que hicieron el desarrollo estarán disponibles para el mantenimiento posterior. Ninguna de las dos organizaciones realiza una adecuada gestión de configuración.
- e) Los proyectos de mantenimiento de software de los departamentos de IT públicos suelen tener una exposición a riesgos bastante más baja que la media del sector IT, al menos, así es considerado por sus responsables.
- f) No es habitual la existencia de procedimientos de aseguramiento de la calidad. Por ejemplo, contradiciendo todas las recomendaciones de búsqueda de calidad, en la organización anónima los encargados de implementar una modificación en un software eran exactamente los encargados de hacer las pruebas correspondientes.
- g) Las pocas técnicas y herramientas utilizadas para la "gestión de eventos", por ejemplo, un repositorio web de eventos y sus documentos asociados, fueron considerados de gran ayuda.
- h) No se ha prestado atención, hasta ahora, a los aspectos centrados en el proceso software (KPA's 3.2 y 3.3).

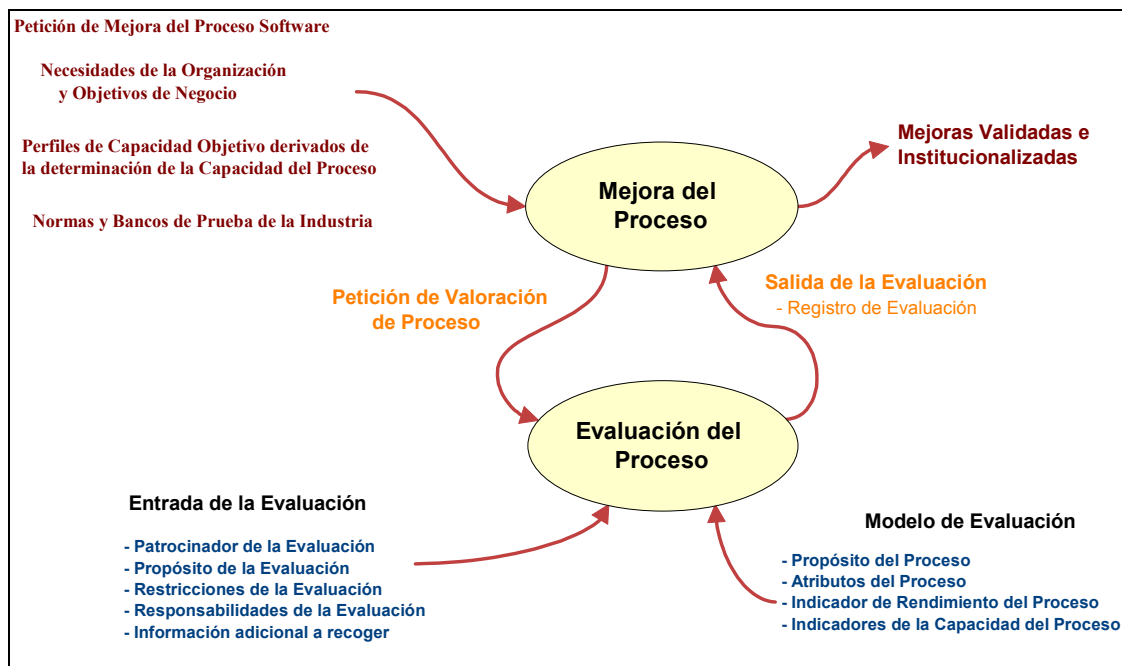


Figura 6-6. Evaluación y mejora de procesos según ISO 15504.

6.3.1.2. Evaluación del Proceso de Mantenimiento.

Para que una organización pueda llevar a cabo una mejora de sus procesos, se hace necesario que previamente se identifiquen los puntos fuertes y débiles de dichos procesos, para lo cual es necesario un proceso de evaluación. En la Figura 6-6 se resume la relación entre estos dos procesos. Como se puede observar en la figura, existe un fuerte acoplamiento entre la evaluación y la mejora. Para poder aplicar un plan de mejora en una organización es necesario desarrollar previamente un modelo de evaluación que permita identificar los aspectos de los procesos que hay que mejorar.

El modelo propuesto en MANTIS para la evaluación y mejora del PMS, aunque sirve para cualquier otro proceso software, está basado en el estándar ISO 15504 (ISO/IEC, 1998f; 1998b; 1998c; 1998d). Esta norma proporciona una guía para realizar la evaluación de un proceso software y un método para la mejora de procesos de una manera continuada. Para ello define un modelo de referencia con las dos dimensiones siguientes (consultar mas detalles en capítulo 3):

- a) *Proceso*, en la que se identifican los objetivos medibles que debe alcanzar cada proceso y su funcionalidad; y
- b) *Capacidad*, que se basa en evaluar un conjunto de atributos asociados a cada proceso para determinar la capacidad del mismo, lo que es necesario para su gestión y mejora.

Nivel Conceptual	Propuestas MANTIS
M3	Modelo MOF (anexo C)
M2	Metamodelos de Proceso Software y de la Medida (capítulo 5)
M1	Modelo de Evaluación y Mejora (este apartado)
M0	Instancias de evaluación y mejora en proyectos concretos del mundo real

Tabla 6-15. La evaluación y mejora del PMS en la arquitectura conceptual de MANTIS.

El modelo propuesto se adapta al marco de trabajo conceptual del Entorno MANTIS (García et al, 2002b), presentado en el capítulo 5, de la siguiente manera (Tabla 6-15): el modelo de proceso de evaluación y mejora pertenece al nivel M1 de la arquitectura conceptual; está basado en el meta-modelo genérico de proceso software de MANTIS (capítulo 5); y, puesto que para evaluar es necesario medir, también se utiliza el meta-modelo de la medida de MANTIS (capítulo 5). Además, en línea con el sistema de procesos de MANTIS (capítulo 5), el proceso de mejora se considera formado por dos grupos de actividades: actividades de evaluación y actividades de mejora.

En el anexo G se muestran las correspondencias M3-M2 y M2-M1 de la arquitectura conceptual de MANTIS para el paquete básico del metamodelo de proceso software. De igual forma, en la Tabla 6-16 se muestran las equivalencias M2-M1 entre los dos metamodelos citados antes (proceso software y medida) y el modelo para las actividades de la fase de evaluación del PMS. La jerarquía de actividades se ha resumido mediante la indexación de las sub-actividades dentro de sus respectivas actividades, aunque en realidad se debería haber indicado mediante interrelaciones del tipo “actividad contiene actividad” (según metamodelo de

los flujos de trabajo del capítulo 5). Para mayor claridad, se han eliminado las clases M2 referidas a procedimientos y recursos, ya que no afectan a la parte central del modelo de evaluación.

Clases M2	Clases M1 (ejemplares M2)
Act Actividad	E - Evaluación del Proceso (grupo de actividades)
	E.1 - Planificar la Evaluación (actividad)
	E.2 - Recolectar datos (actividad)
	E.3 - Validar datos (actividad)
	E.4 - Valoración del proceso (actividad)
	E.5 - Documentación del resultado (actividad)
Age Agente	EVA - Evaluador UO - Unidad organizacional evaluada
Art Artefacto	INPUT - Entrada de la Evaluación
	INPUT.1 – Promotor y organización evaluada
	INPUT.2 – Propósito y objetivos de negocio relacionados
	INPUT.3 – Alcance
	INPUT.3.1 – Proceso a evaluar
	INPUT.3.2 – Niveles de Capacidad incluidos
	INPUT.3.3 – Contexto del proceso
	INPUT.4 – Restricciones y otros aspectos de la evaluación
	OUTPUT - Salida de la Evaluación
	OUTPUT.1 – Datos de realización de la evaluación (fechas, etc.)
	OUTPUT.2 – Aproximación de evaluación empleada
	PP – Perfil del Proceso (ver Figura 6-7)
Contiene-Art-Art Artefacto contiene Artefacto	PLAN – Plan para la evaluación
	Artefacto INPUT contiene artefacto INPUT.1
	Artefacto INPUT.3 contiene artefacto INPUT.3.1
	Artefacto OUTPUT contiene artefacto PP
Desempeña-Age-Rol Agente desempeña Rol	Agente EVA desempeña el rol AC
	Agente EVA desempeña el rol MEE
EsEntradaDe-Art-Act Artefacto es entrada de Actividad	Artefacto INPUT.1 es entrada de actividad E.1
	Artefacto INPUT 3.1 es entrada de actividad E.2
EsResponsableDe-Rol-Act Rol es responsable de Actividad	Rol AC es responsable de actividad E.1
	Rol MEE es responsable de actividad E.5
EsSalidaDe-Art-Act Artefacto es salida de Actividad	Artefacto PLAN es salida de la actividad E.1
	Artefacto PP es salida de la actividad E.4
Incluye-Act-Act Actividad incluye Actividad	Actividad E incluye actividad E.1
	Actividad E incluye actividad E.2
Realiza-Age-Act Agente realiza actividad	Agente EVA realiza actividad E.4

Rol Rol	PRO - Promotor de la evaluación
	AC - Asesor Competente
	MEE - Miembros del Equipo de Evaluación
	...

Tabla 6-16. Modelo para la evaluación y mejora: correspondencias M2-M1.

6.3.1.2.1. Resultados del Proceso de Evaluación.

Quizás lo más interesante del modelo propuesto en MANTIS sea la definición del artefacto de salida (OUTPUT) con los resultados de la evaluación de uno o varios procesos. Para su definición se han utilizado las guías incluidas en las partes 3 (ISO/IEC, 1998f) y 5 (ISO/IEC, 1999) del estándar ISO. En la Figura 6-7 se muestra el diagrama UML que representa la estructura de dicho artefacto tal como se representa en MANTIS. En realidad, su implementación en el repositorio de MANTIS es mediante un DTD con la estructura equivalente (perteneciente al nivel M1) y un documento XML, ajustado a dicho DTD, con los resultados de cada instancia de evaluación concreta.

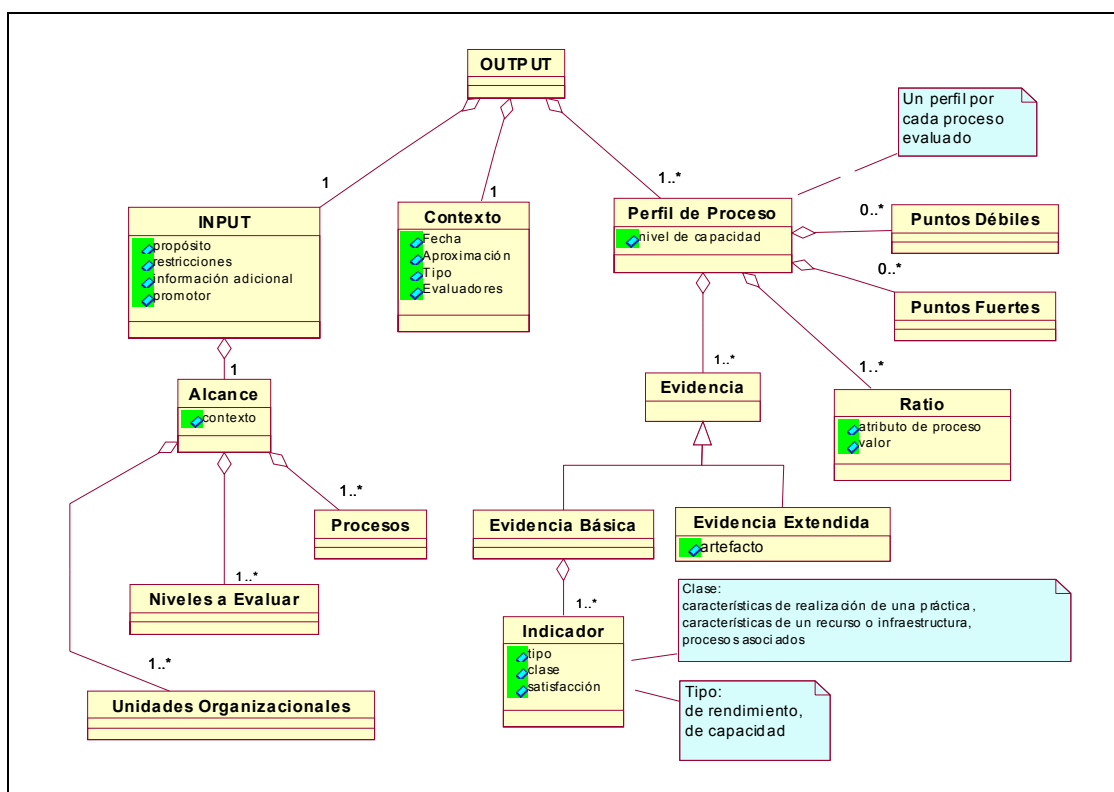


Figura 6-7. Estructura del artefacto OUTPUT con los resultados de la evaluación.

El artefacto OUTPUT está formado por la información de entrada al proceso (artefacto INPUT), información sobre el contexto en el que se ha desarrollado la evaluación y un perfil de proceso (artefacto PP) por cada proceso evaluado. El artefacto de entrada incluye el propósito de la evaluación, las restricciones a las que está sometida, el promotor (persona) que solicita o autoriza la evaluación, el alcance y otra información adicional que pueda ser interesante. El alcance de una evaluación está formado por la información del contexto en el que se solicita la evaluación, el o los procesos sometidos a evaluación, qué niveles de madurez son los que se desean analizar, y cuales son las unidades organizacionales afectadas por los citados procesos. El contexto de la evaluación incluye la fecha, el procedimiento o aproximación seguido, el tipo de evaluación y la relación de evaluadores.

Como puede observarse en la figura anterior, la parte central del artefacto OUTPUT son los sub-artefactos llamados perfiles de proceso (PP). Cada perfil incluye los resultados de la evaluación de uno de los procesos referidos en el INPUT. El principal dato incluido en un

artefacto PP es el nivel de capacidad, que indica en cual de los niveles del modelo de madurez están las unidades organizacionales referenciadas respecto del proceso en cuestión. Además, también se incluyen otras informaciones que completan y justifican dicha evaluación global:

- Los puntos fuertes y débiles detectados en el proceso.
- Los ratios utilizados para determinar el nivel de capacidad. Cada ratio asigna una valoración del grado de cumplimiento (“total”, “mucho”, “a medias”, “poco”, “nada”) de cada uno de los atributos de proceso definidos en el modelo de madurez.
- Las evidencias utilizadas para asignar valoraciones a los ratios. Las evidencias pueden ser extendidas, que se refieren a artefactos tangibles cuya existencia es comprobada, o básicas. Las referencias básicas consisten en una colección de indicadores de proceso cuyo objetivo es intentar reducir la subjetividad en la evaluación.
- El cumplimiento o satisfacción de cada indicador de proceso.

Los indicadores de proceso se distinguen por su utilidad (tipo) y por su naturaleza (clase). Según su utilidad, existen dos tipos: los que sirven para evaluar la realización o rendimiento del proceso (nivel 1 del modelo de madurez) o los que sirven para evaluar la capacidad del proceso (niveles superiores del modelo de madurez). Según su naturaleza, pueden ser de tres clases: características de realización de una actividad, características de un recurso o infraestructura, o procesos asociados que son llevados a cabo.

Por otro lado, gracias a la arquitectura conceptual de MANTIS es posible realizar adaptaciones, no sólo a otros modelos de evaluación (nivel M1), sino también a otros metamodelos (nivel M2) sin necesidad de cambiar los modelos. Por ejemplo, el mismo modelo de proceso de evaluación anterior ha sido representado utilizando SPEM “*Software Process Engineering Metamodel*” (anexo F) como metamodelo del nivel M2 (García et al, 2003a).

6.3.2. Recursos Humanos.

Es conocido que en los proyectos de desarrollo de software se considera que los recursos humanos son los únicos significativos y, por tanto, los costes del resto de tipos de recursos se consideran despreciables. Para los proyectos de mantenimiento se realiza la misma consideración pero, si cabe, con más fuerza ya que las actividades de mantenimiento son muy masivas en utilización de “personal de mantenimiento” (Pigoski, 1997). La mayoría de las técnicas generales y conocidas de gestión de recursos humanos en proyectos informáticos son directamente trasladables a proyectos de mantenimiento, con la salvedad ya comentada. Por esta razón, en el Entorno MANTIS las propuestas específicas para mantenimiento presentadas en el capítulo 4 se han completado, tan solo, con un método para distribuir los recursos humanos cuando una organización tiene que atender a la vez a varios proyectos.

6.3.2.1. Distribución de Recursos Humanos en una Cartera de Proyectos.

Supongamos una organización (mantenedor) que tiene una cartera de proyectos de mantenimiento de software, bien para sí misma o bien porque está prestando un servicio de externalización para otras organizaciones clientes. A continuación se presenta el modelo matemático desarrollado en MANTIS para decidir el reparto de los recursos humanos

disponibles en cada momento entre las peticiones de modificación (PM) de los diversos proyectos.

El modelo está basado en un análisis costes-beneficios suponiendo las siguientes consideraciones:

- Por cada PM atendida, el mantenedor obtiene unos ingresos en función del tiempo-persona que debe dedicar a resolver cada PM.
- Los gastos o costes que soporta el mantenedor por atender cada PM vienen determinados exclusivamente por los costes de los recursos humanos dedicados a tal fin. El coste unitario de los recursos humanos no es constante sino que depende de los días.
- Si el tiempo dedicado sobrepasa un límite establecido previamente (por ejemplo, en un Acuerdo de Nivel de Servicio), entonces a los costes se les tendrá que añadir una penalización que será proporcional al retraso sufrido.
- Sólo se considera un único tipo de recurso humano “personal de mantenimiento”. Es decir, no se distinguen categorías como “codificador”, “programador”, “probador”, etc., cada una con sus propios costes por unidad de tiempo. La razón de no realizar esta consideración es que, siguiendo las directrices marcadas por el método de trabajo de “investigación-acción” (capítulo 2), las organizaciones que formaron el “Grupo Crítico de Referencia” consideraron que esta aportación sólo añadía una complicación bastante grande en la gestión sin que los beneficios les compensaran.

Este modelo también puede aplicarse, con pequeños retoques, a proyectos de desarrollo de software simplemente sustituyendo las PM por sub-proyectos parciales, cuyos resultados o entregables corresponden a los módulos, versiones o “releases” del producto software (Polo et al, 2001a).

6.3.2.1.1. Maximización del Beneficio.

Como cualquier otra organización, el mantenedor debe optimizar los beneficios, es decir, la diferencia entre los ingresos y los costes para todos y cada uno de los proyectos de su cartera:

$$\text{Max}(B) = I - C = \sum_{i=1}^N (I_i - C_i) \quad (6.11)$$

donde B representa el beneficio producido por los N proyectos de la cartera de proyectos; I y C son las funciones de ingresos y costes globales, mientras que I_i y C_i representan los ingresos y costes del proyecto i-ésimo. Como de cada proyecto se suelen tener varias peticiones de modificación (PM) (también se podrían considerar lo mismo para el caso de sub-proyectos), la fórmula se amplía a

$$\text{Max}(B) = \sum_{i=1}^N \sum_{j=1}^{N_i} (I_{i,j} - C_{i,j}) \quad (6.12)$$

y ahora N_i representa el número de PM que tiene el proyecto i-ésimo, mientras que $I_{i,j}$ y $C_{i,j}$ son, respectivamente, los ingresos y costes de la petición j-ésima del proyecto i-ésimo.

Los ingresos de cada petición se pueden calcular mediante la fórmula siguiente:

$$I_{i,j} = \sum_{k=1}^{T_j} (H_{i,j,k} * \alpha_k) \quad (6.13)$$

con los siguientes significados:

- α_k indica el precio que el cliente paga al mantenedor por cada hora de recurso necesaria para servir la petición j-ésima en el día k-ésimo;
- T_j es el número de días totales planeados para realizar la petición j-ésima; y
- $H_{i,j,k}$ representa el número de horas que el mantenedor necesita dedicar el día k-ésimo a la petición j-ésima del proyecto i-ésimo.

Por otro lado, se considera que los costes de cada PM serán diferentes porque para cada una se habrá planificado un número de horas para servirla, los recursos pueden tener distintos precios dependiendo del día (por ejemplo, los días no laborables el precio de los recursos puede aumentar), y puede ser que influyan en los costes de la petición la existencia de retraso en la fecha de entrega o las sanciones por ese retraso. En consecuencia, los costes de la petición j-ésima del proyecto i-ésimo se pueden representar de la siguiente forma:

$$C_{i,j} = R_{i,j} + S_{i,j} \quad (6.14)$$

$S_{i,j}$ indica la cantidad que debe ser pagada al cliente por las sanciones producidas por los retrasos en la petición j-ésima del proyecto i-ésimo, mientras que $R_{i,j}$ representa el coste de los recursos dedicados a la petición j-ésima del proyecto i-ésimo. Su valor se calcula con la fórmula:

$$R_{i,j} = \sum_{k=1}^{T_j} (D_{i,j,k} * \beta_k) \quad (6.15)$$

siendo

- β_k el precio que el mantenedor paga a sus recursos (humanos) por hora el día k-ésimo;
- T_j el número de días planeados para realizar la petición j-ésima; y
- $D_{i,j,k}$ el número de horas que la organización de mantenimiento dedica a la petición j-ésima del proyecto i-ésimo en el día k-ésimo.

La penalización $S_{i,j}$ sufrida por el mantenedor por el retraso en atender la petición j-ésima del proyecto i-ésimo será

$$S_{i,j} = \delta_i * \sum_{k=1}^{T_j} L_{i,j,k} \quad (6.16)$$

con

- δ_i indicando el precio de la penalización que el mantenedor debe abonar por cada día de retraso en la resolución de la PM;
- T_j indicando el número de días planeados para realizar la petición j-ésima; y
- $L_{i,j,k}$ representando el número de días de retraso en acabar la petición respecto de lo previsto.

Una vez que se tienen caracterizadas todas las variables que aparecen en la ecuación (6.12), es posible desarrollar dicha fórmula, obteniendo lo siguiente:

$$\text{Max}(B) = \sum_{i=1}^N \sum_{j=1}^{N_i} (I_{i,j} - R_{i,j} - S_{i,j})$$

y más detalladamente:

$$\text{Max}(B) = \sum_{i=1}^N \sum_{j=1}^{N_i} \left[\left(\sum_{k=1}^{T_j} (H_{i,j,k} * \alpha_k) \right) - \left(\sum_{k=1}^{T_j} (D_{i,j,k} * \beta_k) \right) - \left(\delta_i * \sum_{k=1}^{T_j} L_{i,j,k} \right) \right] \quad (6.17)$$

6.3.2.1.2. Restricciones.

Las variables incluidas en la expresión (6.17) están sujetas a diversas restricciones que pasamos a analizar. La primera restricción viene dada por el hecho de que la suma de las horas de los recursos dedicados durante un día ‘k’ a atender peticiones de modificación no puede ser mayor que el número de horas disponibles de recursos en ese día (que se denota como E_k):

$$\sum_{k=1}^{T_j} (D_{i,j,k}) \leq E_k \quad (6.18)$$

Las siguientes restricciones determinan los valores mínimos de las variables que intervienen:

$$\begin{aligned} D_{i,j,k} &\geq 0 & \forall i \in [1, N] \forall j \in [1, N_i] \forall k \in [1, T_j] \\ H_{i,j,k} &\geq 0 & \forall i \in [1, N] \forall j \in [1, N_i] \forall k \in [1, T_j] \\ \alpha_k &\geq 0 & \forall k \in [1, T_j] \\ \beta_k &\geq 0 & \forall k \in [1, T_j] \\ \delta_i &\geq 0 & \forall i \in [1, N] \end{aligned} \quad (6.19)$$

La última restricción a incluir es necesaria para indicar que el tiempo dedicado a resolver una PM j-ésima del proyecto i-ésimo debe ser igual al tiempo requerido por dicha PM:

$$\sum_{k=1}^{T_j} (D_{i,j,k}) = \sum_{k=1}^{T_j} (H_{i,j,k}) \quad \forall i \in [1, N] \forall j \in [1, N_i] \quad (6.20)$$

6.3.2.1.3. Método de Cálculo.

La función objetivo a maximizar (6.17) no es una función lineal. Tampoco es polinomial debido a los valores aleatorios que pueden tomar los $L_{i,j,k}$. Por tanto, no es posible utilizar el conocido método ‘simplex’ u otros similares y es necesario realizar el cálculo con otro tipo de algoritmos más complejos. En la herramienta CREM (ver capítulo 7) se han implementado dos algoritmos basados en el método devorador, útil para obtener un conjunto de “soluciones factibles” que cumplen unas determinadas restricciones (Horowitz y Sartaj, 1978). Una o varios de dichas soluciones serán óptimas. El algoritmo básico utilizado es el siguiente:

```

procedimiento Devorador (A[1..n])
  solucion= ∅
  desde i=1 hasta n hacer
    si es_factible A[i]
      solucion = solucion union { A[i] }
    fin_si
  fin_desde
  retornar solucion
fin_procedimiento

```

A continuación de aplicar el método devorador, se recorren el conjunto de soluciones factibles encontradas para identificar cual o cuales maximizan la función objetivo. Las dos versiones desarrolladas se resumen en lo siguiente:

- 1) *Minimización de la penalización*: Para cada día, se atienden primero las PM que pertenecen a proyectos con mayor penalización ese día.
- 2) *Minimización de la penalización respecto a los plazos de entrega*: Para atender las PM se tienen en cuenta dos factores: el considerado en el caso anterior, y los días que le faltan a cada PM antes de comenzar a penalizar.

6.3.3. Medida.

En el capítulo 5 ya se comentó la importancia que se concede al proceso de la medida (o la medición) en el Entorno MANTIS. Prueba de ello son la ontología y el metamodelo de la medida propuestos. También se ha explicitado suficientemente en dicho capítulo el modelo del proceso de medición propuesto en MANTIS.

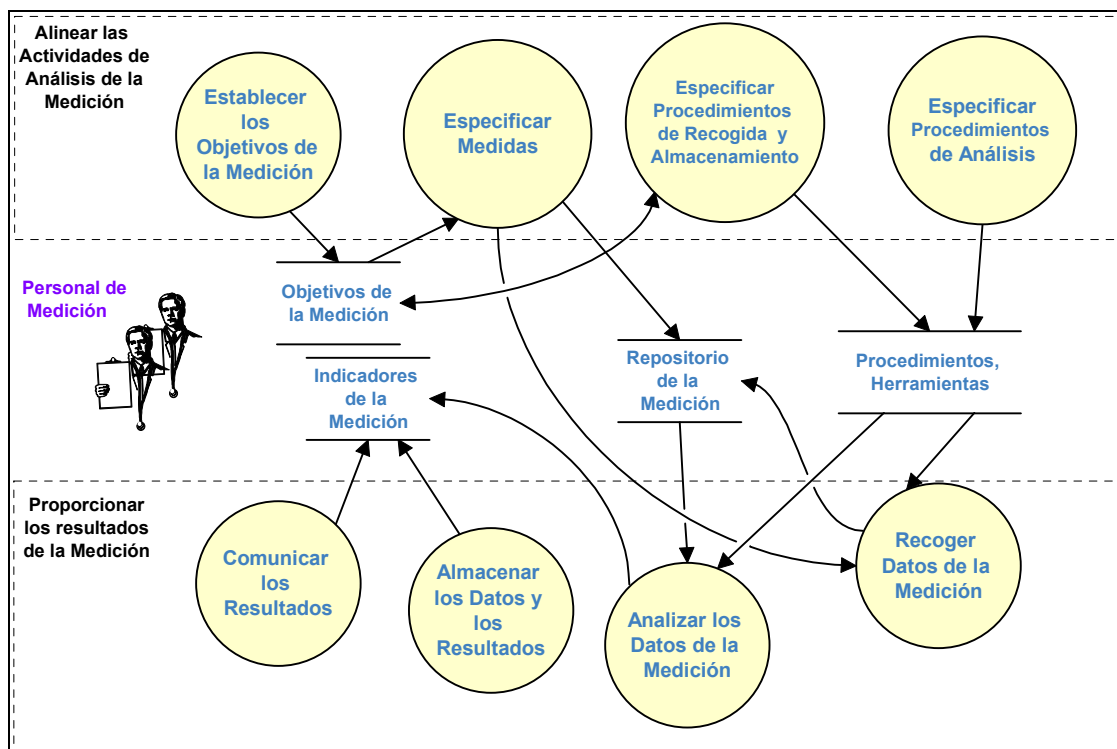


Figura 6-8. El área de proceso clave de "Medición y Análisis" en CMMi.

Las últimas propuestas internacionales para la mejora de procesos software confirman esta importancia de la medición. Así, CMMi (*Capability Maturity Model Integration*) incluye una nueva área de proceso clave denominada “Medición y Análisis” (SEI, 2002). El objetivo de este área es desarrollar y establecer una capacidad de medición que se pueda usar para dar soporte a las necesidades de información de la organización, e implica una ampliación a los conceptos incluidos en el modelo CMM (Figura 6-8). Dicha área da soporte al resto de áreas de proceso proporcionando un marco de trabajo a las organizaciones a la hora de alinear los objetivos y necesidades de medición con un enfoque de medición basado en proporcionar resultados objetivos que sean útiles para la toma de decisiones y acciones correctivas.

Este enfoque del Entorno MANTIS centrado en la medición, confirmado por CMMi con posterioridad, es también consistente con propuestas previas como GQM (*Goal-Question-Metric*) de Basili et al (1994).

La mayoría de los problemas de recolección de datos en un proceso de medición se deben fundamentalmente a una pobre definición de los conceptos y atributos medibles y de las métricas utilizadas (ver ontología de la medida del capítulo 5). Por eso es importante, además de recoger los valores propios de las observaciones del proceso de medición, representar también adecuadamente los metadatos asociados con dichos valores. Esta idea de abstracción es fundamental para poder integrar el proceso de medición de una forma efectiva en una organización. En consecuencia, resulta muy conveniente incorporar un metamodelo genérico para la medición, a partir del cual pueda ser posible derivar modelos concretos de medida que constituyan la base en una organización para sus procesos de evaluación y mejora (ver apartado 6.3.1). Así pues, el metamodelo de la medida presentado en el capítulo 5 constituye una aportación significativa del Entorno MANTIS al proceso de medición.

Otra ventaja de este planteamiento es que es posible diseñar y construir herramientas CASE para la medición de procesos y productos software muy versátiles y generales. Esta es una de las líneas de trabajo abiertas en el Entorno MANTIS, que se comenta en el capítulo 8 (ver propuesta de herramienta GenMETRIC).

Además de las aportaciones de carácter general comentadas, el Entorno MANTIS incluye también algunas propuestas concretas (orientadas al mantenimiento) para la medición:

- Las métricas utilizadas como indicadores de nivel de servicio (apartado 0).
- Las más de 60 métricas incluidas en la metodología MANTEMA, desglosadas por tarea en la que se debe realizar la observación (Polo et al, 1999a).

“Existen dos maneras de realizar un diseño software: hacerlo tan simple que sea obvio que no tiene deficiencias, o hacerlo tan complejo que las deficiencias no sean obvias. El primer método es mucho más difícil” (C.A.R. Hoare).

7. Herramientas Software.

En este capítulo se detallan los elementos del Entorno MANTIS que son herramientas tecnológicas, es decir, los prototipos software desarrollados. Aunque el alcance inicial de esta tesis fue la definición del entorno extendido de ingeniería del software llamado MANTIS y, por tanto, estos prototipos no formaban parte de dicho alcance, se optó por desarrollarlos para probar la factibilidad de las propuestas realizadas, además de por el interés manifestado por las empresas que han colaborado en los proyectos de I+D. De esta manera, se han podido automatizar algunos aspectos de la gestión del mantenimiento que ninguna de las herramientas CASE conocidas abordaba o no lo hacía en la forma adecuada para ser útil en el Entorno MANTIS.

El capítulo está organizado en dos partes diferenciadas. Una primera parte, de carácter general, donde se describen los tipos de herramientas CASE considerados en el Entorno y la arquitectura software que le da soporte; y una segunda parte donde se describen las características y funcionalidad de los principales prototipos desarrollados. Esta segunda parte está organizada en dos categorías que se corresponden con los dos tipos de herramientas CASE internas: horizontales y verticales.

7.1. Introducción.

Como se comentó en el capítulo 5, una de las características principales del Entorno MANTIS es su apuesta por la utilización de herramientas software siguiendo la filosofía de los Entornos de Ingeniería del Software (EIS) integrados (capítulo 3). Para facilitar dicha integración se ha optado por los principios arquitecturales de los EIS orientados a procesos y, más concretamente, por la propuesta conocida como “*Process-centered Software Engineering Environment*” o PSEE (Derniame et al, 1999b), presentada en detalle en el capítulo 3.

En el marco conceptual de MANTIS (ver ontología de los flujos de trabajo del capítulo 5) se prevé la posibilidad de actividades semiautomáticas o automáticas, es decir, actividades que son realizadas por agentes software de forma total o parcial. Dicha posibilidad también está reflejada en los correspondientes metamodelos. Igualmente, al principio del capítulo 5 también se presentaron las características funcionales del Entorno MANTIS en que se ha concretado la filosofía de la propuesta PSEE: utilizar un Sistema de Gestión de Flujos de Trabajo (SGFT) como motor de procesos (*process engine*) para controlar el flujo de información entre los actores o realizadores; incluir un repositorio general de datos de proyectos y metadatos (modelos, metamodelos); y utilizar mecanismos de importación/exportación basados en el uso de formatos estándares abiertos (XML y XMI principalmente).



Las herramientas software presentadas en este capítulo son prototipos que han sido definidos teniendo en cuenta las características y filosofía, ya conocidas, del Entorno MANTIS. En términos de implementación esto ha significado que han tenido que adaptar sus requisitos a la arquitectura software que se presenta en el apartado siguiente. La implementación de estos prototipos ha sido posible gracias a la colaboración de estudiantes de 5º año de ingeniería informática que fueron contratados como becarios de investigación a tiempo parcial y/o que eligieron dicha implementación como proyecto fin de carrera.

7.1.1. Arquitectura Software.

En términos de implementación, el Entorno MANTIS se materializa en un sistema software en el que cada herramienta actúa como un componente software. Según el servicio que prestan, se distinguen dos tipos de componentes, es decir, de herramientas: horizontales y verticales. Además, se contemplan también herramientas externas que se pueden integrar con el sistema por medio de los servicios de importación y exportación.

Los componentes software, en la versión actual de MANTIS, son los siguientes (ver Figura 7-1):

- a) Herramientas Horizontales (servicios de carácter general):
 - a.1) *MANTIS-Tool*: interfaz de integración (apartado 7.2.1).
 - a.2) *RepManager*: gestor del repositorio global de datos y metadatos en formato XML/XMI (apartado 7.2.2).

- a.3) *KM-MANTIS*: gestor de la base de conocimientos (apartado 7.2.3).
- b) Herramientas Verticales (servicios específicos para automatizar algunas actividades):
 - b.1) *MANTOOL*: gestor de peticiones de modificación en base a la metodología MANTEMA (apartado 7.3.1).
 - b.2) *METAMOD*: gestor de modelos y metamodelos (niveles M2 y M1) y sus correspondencias (apartado 7.3.2).
 - b.3) *CREM*: gestor de recursos humanos en una cartera de proyectos (apartado 7.3.5).
 - b.4) *SREM*: estimar el esfuerzo de mantenimiento (apartado 7.3.4).
 - b.5) *MAN-Quest*: gestor de cuestionarios para la evaluación de procesos de mantenimiento (apartado 7.3.6).
 - b.6) *MANTICA*: calcular métricas de mantenibilidad (apartado 7.3.3).

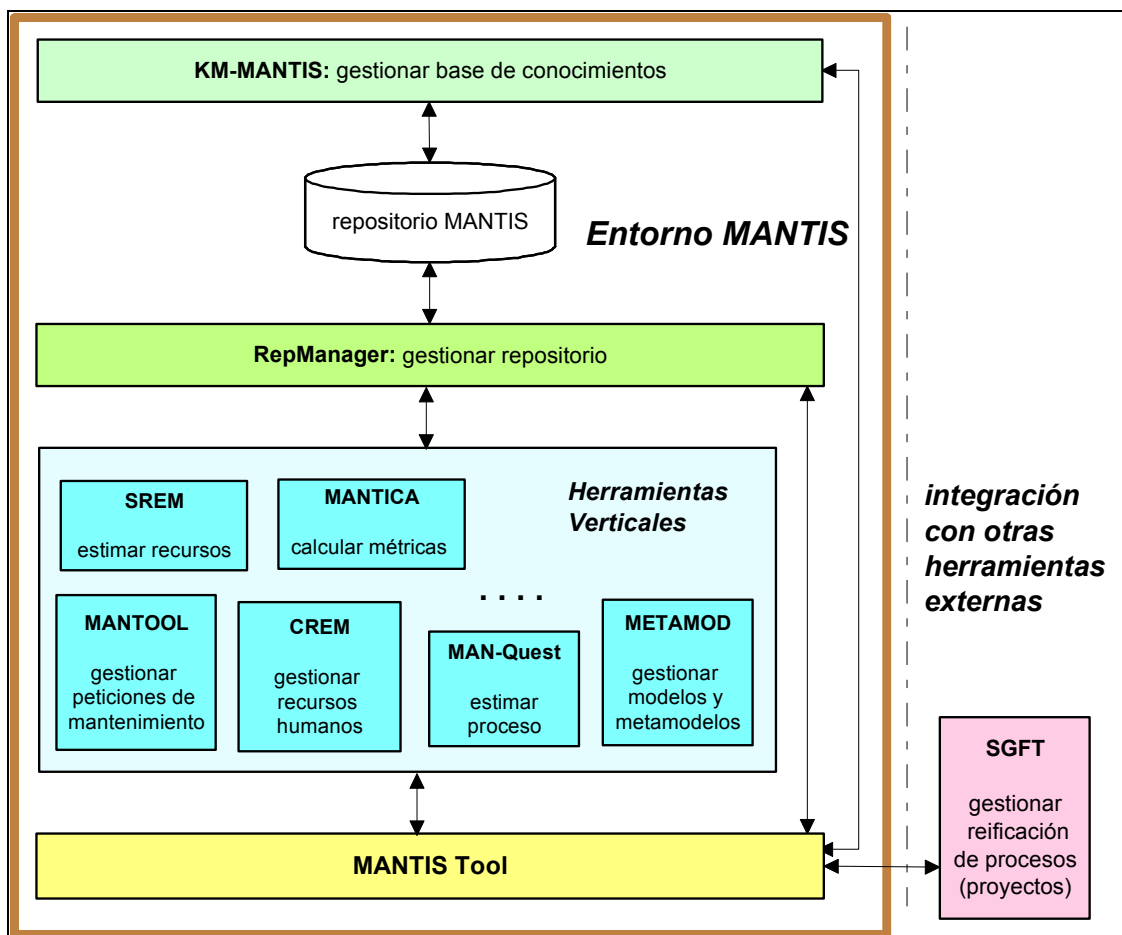


Figura 7-1. Arquitectura software de MANTIS.

En la Figura 7-1 se han representado mediante rectángulos muy alargados las tres herramientas horizontales para representar que sus servicios son útiles para todo el sistema. Una de las líneas de trabajo abiertas en la actualidad es incorporar un nuevo componente horizontal que ofrezca servicios de medición a todo el sistema en base al modelo de medida presentado en el capítulo 5. Este futuro componente, llamado GenMETRIC, está en fase de especificación y

servirá para medir atributos o características tanto de procesos como de productos, tanto de modelos como de su reificación (ver capítulo 8).

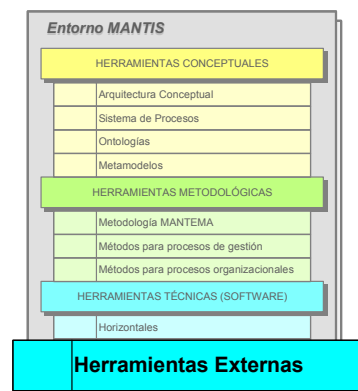
7.1.1.1. Integración de las Herramientas Internas.

La integración de las herramientas de un EIS abarca varias dimensiones diferentes (Wasserman, 1989). En MANTIS estas dimensiones se abordan, con la arquitectura software mencionada, de las siguiente maneras:

- a) *Datos* (habilidad de compartir la información): RepManager ofrece una librería de funciones para utilizar un repositorio global del sistema que puede almacenar cualquier tipo de dato (en formato XML) o metadato (en formato XMI)
- b) *Control* (habilidad de combinar las funcionalidades ofrecidas de forma flexible): Aunque no existe en MANTIS un componente para controlar la ejecución de los proyectos, la arquitectura software permite invocar las funcionalidades aportadas por las diversas herramientas verticales desde una herramienta externa de control de proyectos o de flujo de trabajos. En ese caso, la herramienta MANTIS-Tool hace el papel de intermediaria interpretando las peticiones externas de servicio.
- c) *Presentación* (habilidad de interactuar con las diversas funcionalidades mediante pantallas de apariencia similar y modos de interacción similares): Todas las herramientas interactivas de MANTIS utilizan un interfaz gráfico de usuario normalizado (en ambiente Windows). Adicionalmente, MANTIS-Tool incorpora un interfaz de integración desde el cual el usuario puede invocar a las demás herramientas interactivas.
- d) *Procesos* (habilidad de acceder a las funcionalidades utilizando un proceso software predefinido): Todas las herramientas verticales respetan el sistema de procesos de MANTIS (capítulo 5). Esto se concreta en que el interfaz siempre indica el proceso, subprocesos, actividades, etc, en el que se sitúa el usuario en cada momento, y el modelo y/o metamodelo utilizado.
- e) *Marco de Trabajo*: Todas las herramientas están definidas de acuerdo con el marco conceptual presentado en el capítulo 5.

7.1.1.2. Integración de Herramientas Externas.

La manera de integrar herramientas externas en el Entorno MANTIS es utilizando las funcionalidades del gestor del repositorio (ver apartado 7.2.2) para la exportación y la importación de datos y metadatos en formato XML/XMI. Además, de cara a una futura versión de MANTIS, se está analizando la posibilidad de realizar dicha interacción mediante la tecnología de “servicios web”. También se está estudiando generalizar esta nueva forma de interacción a todas las herramientas del entorno mediante el uso del estándar WSDL (*Web Services Description Language*) del W3C (2003) como forma de definir, almacenar y compartir los servicios EIS (ver capítulo 3) que cada una de las herramientas del Entorno ofrece al resto de herramientas o a los usuarios. Las dos clases principales de herramientas externas contempladas en MANTIS son los “Sistemas de Gestión



de Flujos de Trabajo” (SGFT) y los “Servidores de Documentos XML”. Los primeros se comentan a continuación y los segundos se proponen como soporte a utilizar por el gestor de la base de conocimientos (apartado 7.2.3).

7.1.1.2.1. Uso de Sistemas de Gestión de Flujos de Trabajo.

El marco conceptual del Entorno MANTIS asigna un papel importante a los Flujos de Trabajo (FT). En el capítulo 5 ya se presentaron y justificaron suficientemente las razones de esta decisión. Esta importancia se concreta y resume en:

- a) La *ontología de los FT* para representar la descomposición de las actividades complejas en otras más simples, para definir las restricciones temporales entre unas y otras actividades, y para controlar el estado de realización de las actividades y de los proyectos.
- b) El *paquete de los FT*, incluido dentro del metamodelo genérico de proceso software, para definir los diversos modelos de FT que representan los procesos (de negocio) de una organización mantenedor.

En el ámbito de las herramientas software, la propuesta del Entorno MANTIS también asigna una importancia especial a los FT. De hecho, el principal tipo de herramienta externa prevista en MANTIS son los SGFT, ya que la arquitectura de MANTIS propone delegar en uno de ellos la responsabilidad sobre la reificación de los procesos, es decir, el seguimiento y captura de los datos referidos a lo que ocurre durante la realización de los proyectos de mantenimiento (Ruiz et al, 2001a; 2001b). La razones de esta propuesta se resumen en el siguiente postulado básico:

“La tecnología actual de Flujos de Trabajo, y en concreto los Sistemas de Gestión de Flujos de Trabajo, es útil para gestionar los procesos de desarrollo o mantenimiento del software desde una perspectiva de proceso de negocio, más amplia que la meramente de ingeniería del software”.

Este postulado está basado en las recientes experiencias de diversos autores sobre la utilidad de los FT para el modelado de procesos software y la utilidad de los SGFT para la reificación de dichos procesos software (Ocampo y Botella, 1998), (Canós et al, 1999), (Aversano et al, 2001). La opinión del autor de esta tesis es que intentar desarrollar herramientas para la reificación de procesos software (como postulan, o postulaban hasta hace poco, algunos proyectos de desarrollo de EIS) en vez de utilizar la tecnología, ya madura, de los SGFT es caer en el error de querer inventar la rueda o, al menos, de realizar un esfuerzo baldío e innecesario.

En el uso de SGFT existen dos fases claramente diferenciadas: Diseño (*Design-Time*), referida al modelado conceptual del FT, y Ejecución (*Run-Time*), referida a la reificación del FT. Igualmente, en MANTIS, el ciclo de vida de un proyecto concreto y real de mantenimiento de software tiene ambas fases:

- a) En la fase de diseño se realiza el modelado conceptual de los procesos mediante la especificación de "sus" FT asociados. Dichos modelos de proceso se implementan en el nivel M1 de la arquitectura conceptual, para lo cual se puede utilizar la herramienta METAMOD (apartado 7.3.2). También es posible utilizar otras herramientas de diseño conceptual, bien específicas para flujos de trabajo de procesos (como las incluidas en

algunos SGFT) o bien generales (como Rational Rose ©), siempre y cuando dichas herramientas tengan la opción de importar y exportar definiciones en formato XMI/XML ²⁶. Estos modelos de proceso se almacenan en el repositorio de MANTIS utilizando los servicios del componente RepManager (apartado 7.2.2). Cuando los modelos de proceso se han diseñado utilizando programas externos a MANTIS (distintos de METAMOD), además de su importación al repositorio de MANTIS, también es necesario realizar su integración en la arquitectura de MANTIS, lo cual consiste en asignar correspondencias con los correspondientes metamodelos del nivel M2. Esta última tarea, más que la de servir para el diseño de los modelos de proceso, es realmente la razón de la existencia de la herramienta METAMOD.

- b) En la fase de ejecución se realiza la reificación de los citados modelos de proceso. Por las razones ya comentadas, y siguiendo el postulado básico anterior, en MANTIS esta tarea se delega en un SGFT externo. Para ello, el SGFT debe poder interactuar con las herramientas internas (verticales u horizontales) de MANTIS, al igual que con cualquier otra herramienta software externa a MANTIS.

En la Figura 7-2, una adaptación de la propuesta de Canós y Penadés (2000), se resumen ambas fases y los conceptos que, en consonancia con las ontologías de MANTIS, intervienen en ellas.

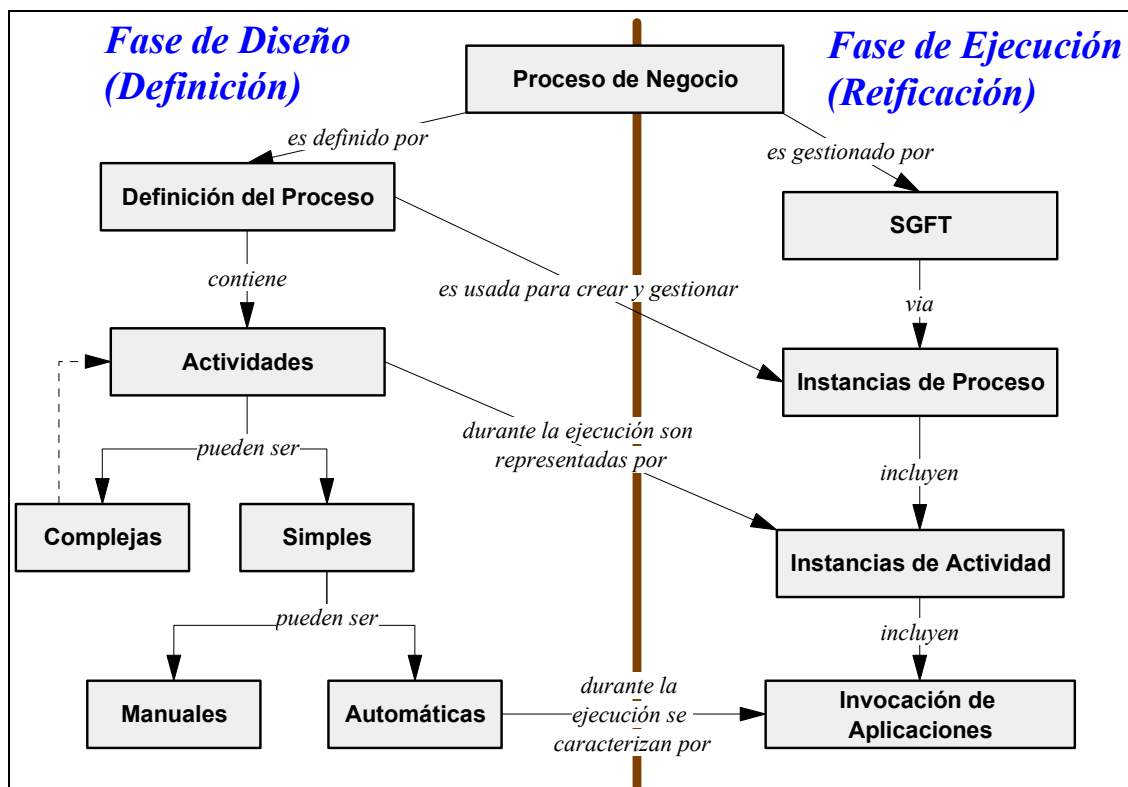


Figura 7-2. Las fases de diseño y de ejecución.

²⁶ Las versiones actuales de “Rational Rose” y de algunos SGFT conocidos, como “Ultimus Workflow”, incluyen la posibilidad de importación/exportación en XMI y/o XML. El uso de estas herramientas tiene la ventaja, frente a METAMOD, de poder utilizar entornos gráficos de diseño muy elaborados.

En términos del modelo de servicios para EIS del estándar ISO 15940 (capítulo 3), el papel desempeñado para el Entorno MANTIS por dicho SGFT es que presta los servicios de iniciación, utilización y supervisión de procesos. En términos de la propuesta PSEE (Derniame et al, 1999b), dicho rol corresponde al denominado “gestor de procesos” (ver PSEE en capítulo 3). Este papel del SGFT es posible gracias a sus componentes denominados, según el modelo de referencia de la WfMC (1995), “servicio de reificación de FT” (*workflow enactment service*) y “motor de FT” (*workflow engine*)²⁷.

Para la interacción entre MANTIS y el SGFT, que realiza y gestiona la reificación de los procesos, es necesario satisfacer los dos requisitos siguientes:

- 1) Incluir en MANTIS el metamodelo de los FT utilizado por el SGFT.
- 2) Utilizar un formato de intercambio, es decir, de envío de mensajes compatible.

En cuanto al primer requisito, en los últimos años se ha producido una estandarización de la tecnología de SGFT que ha estado liderado por la “*Workflow Management Coalition*” (WfMC). Es por ello que en el marco conceptual de MANTIS se ha apostado por el modelo de referencia de FT propuesto por dicha organización (WfMC, 1995), completado por las propuestas de Sadiq y Orłowska (1996) referidas a la estructura de los diagramas de flujo utilizados, que no es detallada en este estándar. Como consecuencia de lo anterior, el metamodelo de FT propuesto en el capítulo 5, perteneciente al nivel conceptual M2, se adapta a esta norma de la WfMC.

Respecto del segundo requisito, la clave está en usar formatos de intercambio estandarizados. En el Entorno MANTIS dichos formatos son XML para datos y XMI para metadatos (ver anexo I para ejemplos de documentos). Así, en MANTIS se ha incluido un documento XMI que representa el modelo de referencia de FT de la WfMC. Además, los modelos de FT concretos se representan también como documentos XMI, del nivel M1, que satisfacen el documento XMI anterior. Esta satisfacción se refiere a que el documento XMI del nivel M2 es, realmente, el DTD de los documentos XMI del nivel M1.

Durante la realización de este trabajo se han llevado a cabo pruebas prácticas de interacción entre MANTIS y la herramienta “Ultimus Workflow” (versión de evaluación). Estas pruebas han consistido en la importación y exportación en ambas direcciones de modelos y metamodelos (niveles M1 y M2) mediante archivos XMI/XML.

Recientemente, la WfMC ha aprobado un nuevo estándar de lenguaje basado en XML para representar procesos en un SGFT, llamado “*Workflow Process Definition Interface - XML Process Definition Language (XPDL)*” (WfMC, 2002). En estos momentos se está diseñando un nuevo componente software de MANTIS para ofrecer servicios de importación y exportación utilizando este lenguaje. Este componente ofrecerá dos funcionalidades principales: importar/exportar modelos de proceso usando XPDL, e importar/exportar en XML datos de ejecución de proyectos basados en uno o varios modelos de proceso definidos previamente en XPDL. La primera opción se integra en la arquitectura de MANTIS como correspondencias

²⁷ Según la WfMC (1995), el “Servicio de Reificación de FT” es un servicio software que consta de uno o varios motores de FT para crear, gestionar y ejecutar instancias de FT. Un “Motor de FT” es un servicio software que provee un entorno *run-time* para la ejecución de instancias de FT.

M2-M1 útiles para la fase de diseño, mientras que la segunda se refiere a correspondencias M1-M0 útiles para la fase de ejecución.

En los párrafos anteriores se ha comentado cómo se “ve” el SGFT externo desde el punto de vista del sistema MANTIS. Presentamos a continuación dicha relación desde el punto de vista contrario: cómo el SGFT “ve” al sistema MANTIS. Para el SGFT, MANTIS es una típica aplicación cliente, tal como son definidas en el modelo de referencia de la WfMC (1995), es decir, es un “manejador de listas de trabajo”. Para la WfMC, una lista de trabajo (*worklist*) es una cola de ítems de trabajo (es decir, lo que en las ontologías de MANTIS y sus metamodelos asociados son actividades simples) asignados a un usuario particular del SGFT por el servicio de reificación de FT. Este modelo de referencia establece cuatro formas diferentes de interacción: basada en host, con repositorio compartido, mediante correo electrónico o similar, y mediante paso de mensajes. La arquitectura presentada del sistema MANTIS permite utilizar dos de los cuatro mecanismos (ver Figura 7-3):

- Repositorio compartido*, consistente en utilizar el repositorio XML/XMI de MANTIS como repositorio compartido por el SGFT. Para ello es necesario que el SGFT ofrezca un servicio para invocar llamadas a API's de programación como la ofrecida por la herramienta RepManager.
- Paso de mensajes*: Esta opción se facilitará cuando esté disponible el nuevo componente de MANTIS, ya comentado, para comunicar utilizando el lenguaje XPDL. En su defecto, se puede recurrir a los servicios de exportación/importación, tanto de MANTIS como del SGFT, utilizando archivos XMI/XML intermedios como medio físico para la comunicación.

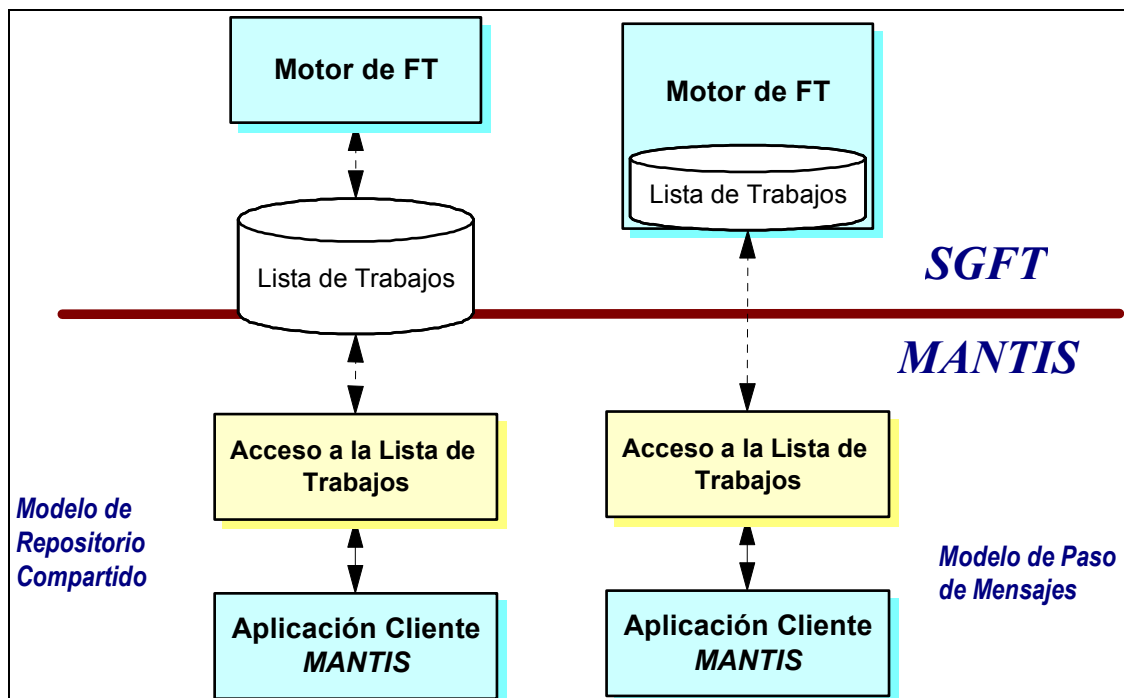
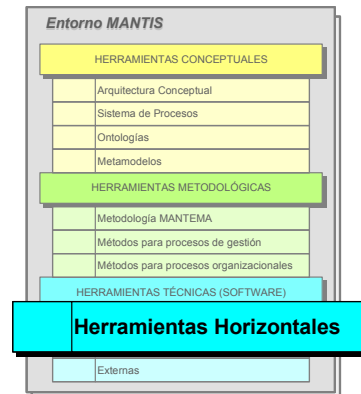


Figura 7-3. Modelos de interacción de MANTIS con un SGFT externo.

7.2. Herramientas Horizontales.

Estas herramientas facilitan servicios de utilidad general para todos los demás componentes del Entorno. Son las siguientes: el interfaz de integración MANTIS-Tool, el gestor del repositorio RepManager y el gestor de la base de conocimientos KM-MANTIS. Por su especial significación y relación directa con el marco conceptual del Entorno MANTIS, la herramienta RepManager será presentada mucho más en detalle que las otras dos.



7.2.1. Interfaz de Integración: MANTIS-Tool.

Tal como muestra la Figura 7-1, MANTIS-Tool es un componente horizontal del Entorno que hace de intermediario entre el resto de componentes, especialmente el gestor del repositorio, y las aplicaciones externas como el SGFT que actúa de motor de procesos. Para ello, esta herramienta se limita a re-direccionar las llamadas, que recibe desde los demás componentes o desde las aplicaciones externas, a la aplicación adecuada.

La segunda funcionalidad de MANTIS-Tool va destinada a prestar el mismo servicio anterior pero para los usuarios humanos del sistema (Ruiz y Piattini, 2001). Esto consiste, sencillamente, en una capa de presentación interactiva desde la cual es posible invocar los diversos componentes de MANTIS. Adicionalmente, esta capa interactiva sirve también de guía y ayuda para los usuarios del sistema MANTIS ya que les permite tener una visión de conjunto de los diversos componentes. La metáfora visual de esta capa interactiva está basada en el concepto de proceso. Más concretamente, todo se le muestra al usuario siguiendo el sistema de procesos de MANTIS presentado en el capítulo 5. Además, la nomenclatura y conceptos utilizados siempre son los indicados en las ontologías.

7.2.2. Gestión del Repositorio: RepManager.

El gestor del repositorio es un componente central del sistema software del Entorno MANTIS. RepManager está diseñado para poder almacenar cualquier dato o metadato de interés para la gestión de proyectos de mantenimiento de software. Estos datos y metadatos consisten, principalmente, en los siguientes (Ruiz et al, 2002a):

- *Datos de producto*: código fuente, versiones, datos para gestión de la configuración, documentación, ejecutables, conjuntos de pruebas, resultados de pruebas, etc.
- *Datos de proceso*: definición explícita de metamodelos y modelos de procesos, información de estado de la reificación de procesos, datos para análisis y evolución de procesos, datos históricos, datos de gestión de proyectos, etc.
- *Datos organizacionales*: propietarios de productos de trabajo, roles y responsabilidades, información de los equipos de trabajo, datos de gestión de los recursos, etc.

Los requisitos funcionales que se definieron para la implementación de RepManager fueron los siguientes:

- Será un componente software autónomo con soporte para documentos XML en general mediante la utilización de la API de programación “*Document Object Model*” (DOM) del W3C (1998b).
- Sus servicios deben poder ser invocados desde METAMOD o desde las demás herramientas de forma interactiva o *batch*.
- El formato para representar los datos, correspondientes al nivel M0 de la arquitectura conceptual, es el *Extensible Markup Language* (XML) del W3C (1998a).
- Para representar metadatos (niveles M1 y M2) se utiliza el lenguaje *XML Metadata Interchange* (XMI) de la Object Management Group (OMG, 2000b).
- Debe incorporar servicios para importar y exportar modelos (nivel M1) y metamodelos (nivel M2) en formato XMI.
- El servicio de importación debe capturar en formato XMI los modelos y metamodelos definidos por cualquier herramienta de metamodelización. Su representación interna estará basada en la jerarquía de clases propuesta por la especificación *Meta Object Facility* (MOF) del OMG (2002b).
- El servicio de exportación debe representar los modelos y metamodelos almacenados en el repositorio en archivos en formato XMI, siguiendo el procedimiento contrario al de importación.

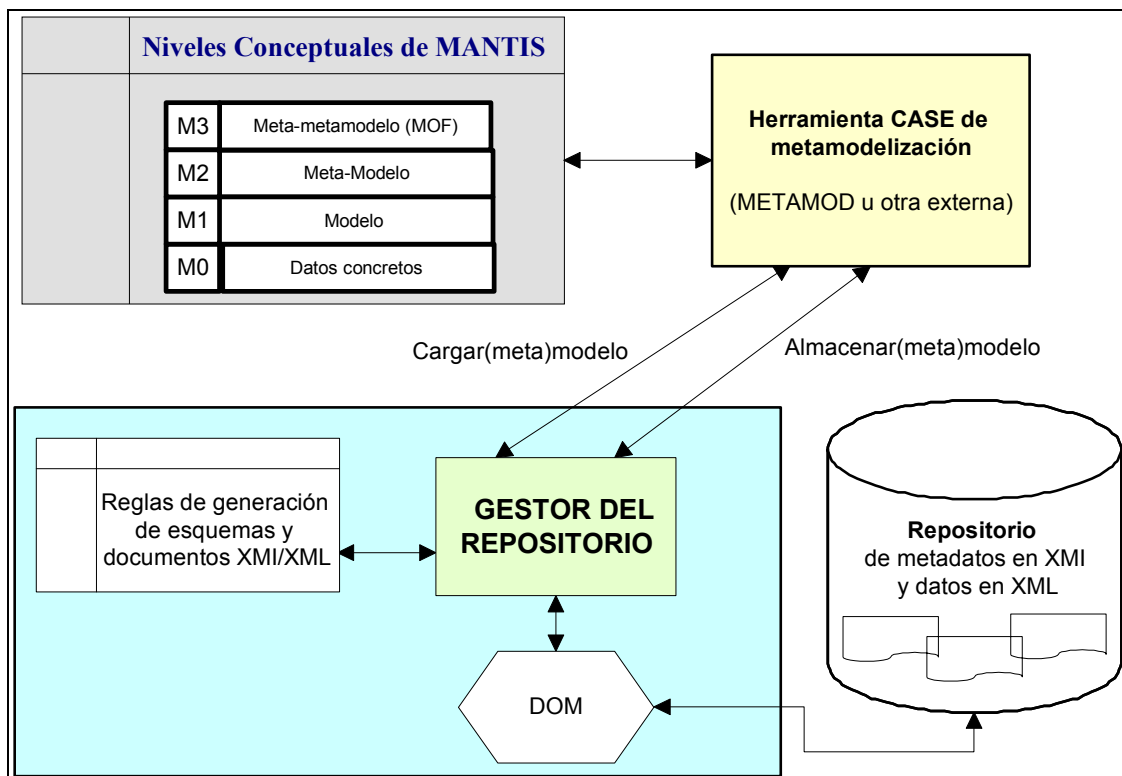


Figura 7-4. Esquema de funcionamiento de RepManager.

La Figura 7-4 muestra el esquema básico de funcionamiento de RepManager y su relación con las herramientas de metamodelado (HMM). Como se observa en esta figura, el gestor proporciona a las HMM dos servicios básicos: almacenamiento de modelos y metamodelos, y exportación/importación de dichos modelos y metamodelos (Ruiz et al, 2002c), (García et al, 2001b). Las HMM invocan estos servicios proporcionando al gestor toda la información necesaria, contenida en un conjunto de objetos agrupados en clases según se detalla más adelante, para la generación de los documentos XMI que representan las correspondencias entre los distintos niveles y los DTD que representan cada tipo de documento XMI generado. Para la generación de documentos XMI y sus DTD's asociados el gestor utiliza las reglas de generación especificadas en el estándar XML. Dichas reglas de generación están definidas en forma de reglas EBNF (*Extended Backus Naur Form*), desde el punto de vista de representación, y en OCL (*Object Constraint Language*) en términos de implementación. Por esta razón, la implementación de la utilidad de generación de documentos del gestor está basada en la implementación especificada en OCL. Finalmente, para la manipulación del repositorio, es decir, para el almacenamiento y recuperación de la información a nivel de documento, se usa la API de programación DOM. Esta API proporciona una colección de clases que representan la estructura jerárquica en que se basa todo documento XML. El modelo proporciona clases que representan documentos, nodos, listas de nodos, etc., con las propiedades y métodos necesarios para construir documentos XML basándose en la estructura jerárquica que los caracteriza.

Una opción alternativa a DOM sería utilizar un servidor de documentos XML pero esto tiene el problema de que el núcleo básico del Entorno MANTIS, su repositorio, dependería de una herramienta externa.

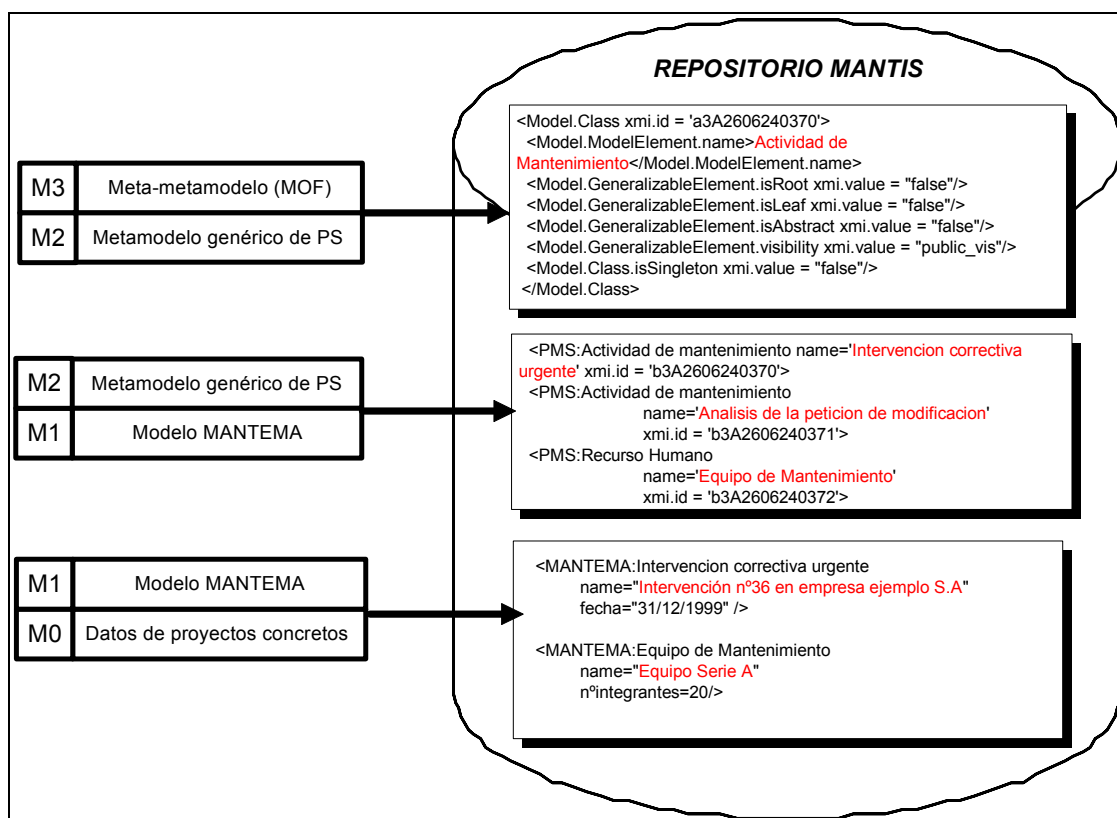


Figura 7-5. Ejemplos de los tres tipos de documentos XMI almacenados en RepManager.

Para ayudar a gestionar de forma integrada los proyectos de mantenimiento, es necesario que el Entorno MANTIS permita la integración y manejo de todos los datos y metadatos correspondientes a los cuatro niveles especificados en la arquitectura conceptual del Entorno. Para ello, todo documento XMI/XML almacenado en el repositorio representa una correspondencia entre un nivel M_i y un nivel M_{i-1} . (Ruiz et al, 2002b). Es decir, en todo documento XMI/XML existen metadatos del nivel M_i que describen los datos del nivel M_{i-1} correspondientes, que son instancias de dichos metadatos. Por tanto, teniendo en cuenta los cuatro niveles conceptuales de MANTIS, en el repositorio se almacenan tres tipos de documentos, dos en formato XMI y uno en formato XML sin XMI (ver Figura 7-5):

- Documentos XMI para las correspondencias M3-M2 como, por ejemplo, entre el modelo MOF (anexo C) y el metamodelo genérico de proceso software (capítulo 5). El DTD que representará este tipo de documentos XMI es único y corresponde al DTD del modelo MOF (que es el único meta-metamodelo del nivel M3 considerado en MANTIS).
- Documentos XMI para las correspondencias M2-M1 como, por ejemplo, entre el metamodelo genérico de proceso software y el modelo concreto de PMS de la metodología MANTEMA (anexo B). En este caso el DTD representa el metamodelo del nivel M2. Para el ejemplo sería el DTD del citado metamodelo genérico de proceso software.
- Documentos XML para las correspondencias M1-M0 como, por ejemplo, entre el modelo concreto de PMS de la metodología MANTEMA y los datos de un proyecto particular aplicado en una empresa que sigue dicha metodología. En este caso el DTD representa el modelo del nivel M1. En el ejemplo el DTD sería el del modelo de PMS de MANTEMA.

La implementación del prototipo actual de RepManager se realizó como proyecto fin de carrera por el estudiante Félix García (García, 2001).

Para mejor comprensión de las características de RepManager, en la presente tesis se han incluido estas otras informaciones adicionales:

- En los apartados siguientes se presentan, con algo más de detalle, los casos de uso definidos y la arquitectura interna, incluyendo sub-componentes.
- En el anexo H se incluye un resumen del manual de utilización de este componente, indicando la API que incorpora.
- En el anexo I se muestran ejemplos de algunos de los documentos generados y almacenados por RepManager. Para que el lector pueda comparar mejor, se han incluido los XMI's (de los niveles M2 y M1) y el archivo DTD, correspondientes a los ejemplos de metamodelo y modelo del anexo G.

7.2.2.1. Casos de Uso.

El único actor a considerar es una “Herramienta de metamodelización” (HMM), ya sea METAMOD u otra externa a MANTIS como *Rational Rose* o un SGFT, que utilice MOF para definir sus modelos y metamodelos. A continuación se explican, con más detalle, cada uno de los casos de uso mostrados en la Figura 7-6:

- a) *Importar*: La HMM activa esta funcionalidad cuando desea cargar un documento XMI, que contendrá un modelo (nivel M1) asociado a un metamodelo, o un metamodelo (nivel M2) asociado a un meta-metamodelo (que en MANTIS siempre es el modelo MOF). Los pasos a realizar en la consecución del caso de uso son:
- a.1) La HMM selecciona un documento XMI a cargar indicando si la sintaxis del documento debe ser validada con respecto a un DTD, e indicando el nivel de la arquitectura MOF de los contenidos del mismo.
 - a.2) Si el nivel especificado es 2 (un metamodelo), se inicia el caso de uso “Importar Metamodelo”. Si el nivel especificado es 1 (un modelo), se inicia el caso de uso “Importar Modelo”.

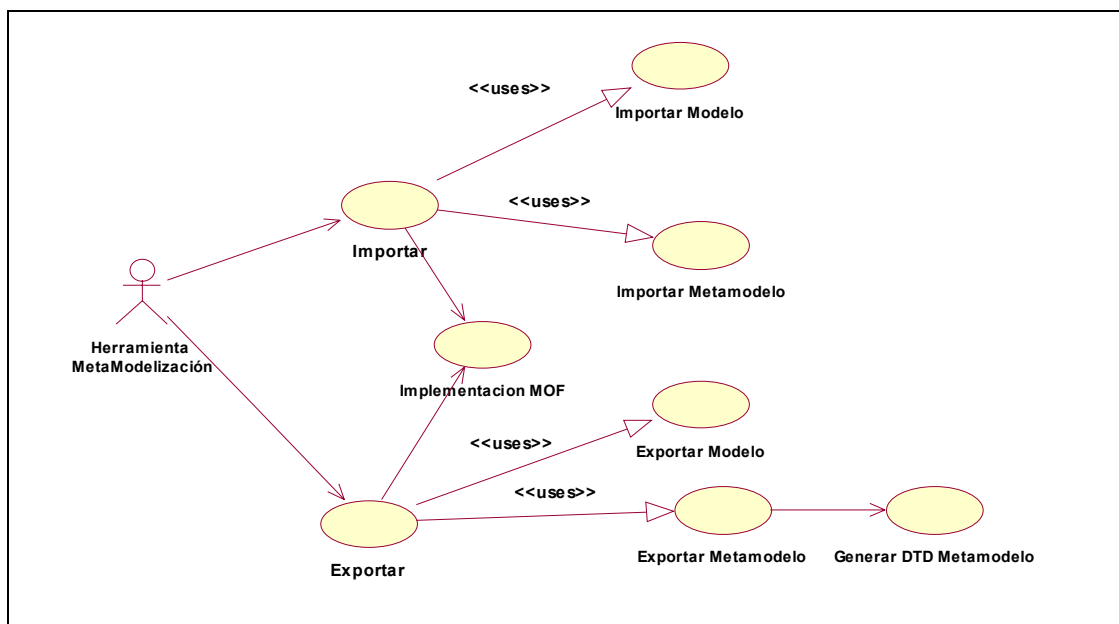


Figura 7-6. Diagrama UML de casos de uso de RepManager.

- b) *Importar metamodelo*: Este caso de uso es iniciado por el caso de uso Importar cuando la HMM selecciona un documento XMI a cargar que contiene información sobre un metamodelo. Como resultado, la HMM recibe el metamodelo contenido en el documento y representado según las clases que forman el núcleo del modelo MOF. Los pasos son:
- b.1) Verificar que el nivel especificado en la carga es 2 (metamodelo).
 - b.2) Se realiza la carga correspondiente, creando los objetos necesarios y teniendo en cuenta si se debe verificar la sintaxis del documento que representa el metamodelo con respecto al DTD que lo representa. Todos los metamodelos a importar son instancias del modelo MOF, por ello el DTD de todos los metamodelos a manejar será el DTD del modelo MOF.
 - b.3) Se devuelve el metamodelo creado a partir del documento.
- c) *Importar modelo*: Este caso de uso es iniciado por el caso de uso Importar cuando la HMM selecciona un documento XMI a cargar que contiene información sobre un modelo. Como resultado de la ejecución de este caso de uso, la HMM recibe el modelo contenido en el

documento y representado según especifica el estándar MOF y, en particular, según su interfaz reflectiva. Los pasos son:

- c.1) Verificar que el nivel especificado en la carga es 1 (modelo).
 - c.2) Se realiza la carga correspondiente, creando los objetos necesarios y teniendo en cuenta si se debe verificar la sintaxis del documento que representa el modelo con respecto a un DTD (que contendría la descripción del metamodelo del nivel M2 del que el modelo es instancia).
 - c.3) Se devuelve el modelo creado a partir del documento.
- d) Exportar: La HMM activa esta funcionalidad cuando desea crear un documento XMI, que contendrá un modelo (nivel M1) asociado a un metamodelo, o un metamodelo (nivel M2) asociado a un meta-metamodelo (que en MANTIS siempre es el modelo MOF). Los pasos a realizar en la consecución del caso de uso son:
- d.1) La HMM indica el nombre y la ruta correspondiente al documento XMI a generar y el nivel de la jerarquía MOF correspondiente a los contenidos del documento.
 - d.2) Si el nivel especificado es 2 (un metamodelo), iniciar caso de uso “Exportar Metamodelo”. Si el nivel especificado es 1 (un modelo), iniciar caso de uso “Exportar Modelo”.
- e) Exportar metamodelo: Este caso de uso es iniciado por el caso de uso Exportar cuando la aplicación selecciona un documento XMI a generar que contiene información sobre un metamodelo. Como resultado, se crea y almacena en el repositorio dicho documento. Este caso de uso también puede generar, de forma opcional, un DTD asociado al metamodelo. Los pasos son:
- e.1) Verificar que el nivel especificado es 2 (metamodelo).
 - e.2) Se realiza la creación y almacenamiento del documento XMI correspondiente. Opcionalmente se puede requerir que se cree un DTD que represente el metamodelo. Si se especifica esta opción, se activa el caso de uso “Generar DTD Metamodelo”.
- f) Exportar modelo: Este caso de uso es iniciado por el caso de uso Exportar cuando la aplicación selecciona un documento XMI a generar que contiene información sobre un modelo. Como resultado, se crea y almacena en el repositorio dicho documento. Los pasos son:
- f.1) Verificar que el nivel especificado es 1 (modelo).
 - f.2) Se realiza la creación y almacenamiento del documento XMI correspondiente.
- g) Generar DTD metamodelo: Este caso de uso es iniciado por el caso de uso “Exportar Metamodelo” cuando es requerido por la aplicación. Para la generación recoge la información del metamodelo y produce como resultado un documento de texto correspondiente al DTD. Los pasos son:
- g.1) Verificar que el caso de uso “Exportar Metamodelo” incluye la opción de activar el caso “Generar DTD metamodelo”.
 - g.2) Se realiza la creación y almacenamiento del documento DTD correspondiente.
- h) Implementación MOF: Este caso de uso es de alto nivel y recoge toda la funcionalidad de un sub-componente software que implementa la especificación MOF. A través de dicha especificación, toda HMM puede gestionar los modelos y metamodelos necesarios de forma robusta y eficiente (ver apartado 7.2.2.3). La interacción del gestor del repositorio con este

componente es fundamental, ya que proporciona el enlace de comunicación entre dicho gestor de repositorio y cualquier HMM que quiera hacer uso del estándar MOF para representar sus metamodelos. A través del mismo, el gestor puede recoger la funcionalidad sobre modelos y metamodelos para la generación de documentos XMI y crear los elementos necesarios para representar de forma adecuada la información sobre modelos y metamodelos contenida en los documentos XMI.

7.2.2.2. Arquitectura de Tres Capas.

RepManager, al igual que las otras herramientas software internas de MANTIS, tiene una arquitectura interna con las tres capas típicas de presentación o interfaz, dominio, y persistencia o almacenamiento. En el diagrama de tres niveles mostrado en la Figura 7-7 se puede observar que la clase GestorRepositorio es la que proporciona la interfaz con el gestor. Por tanto, sólo a través de dicha clase es posible la comunicación del gestor con otras aplicaciones. A través de la interfaz las aplicaciones indican la operación a realizar y se pasa la información a la clase responsable en la capa de dominio de cumplir con la funcionalidad encargada. A nivel de almacenamiento se hace uso de DOM para la construcción de los documentos XML y de un agente de ficheros de texto que se encarga de la generación de los DTD's y, en caso de error en la importación o exportación, del fichero de errores.

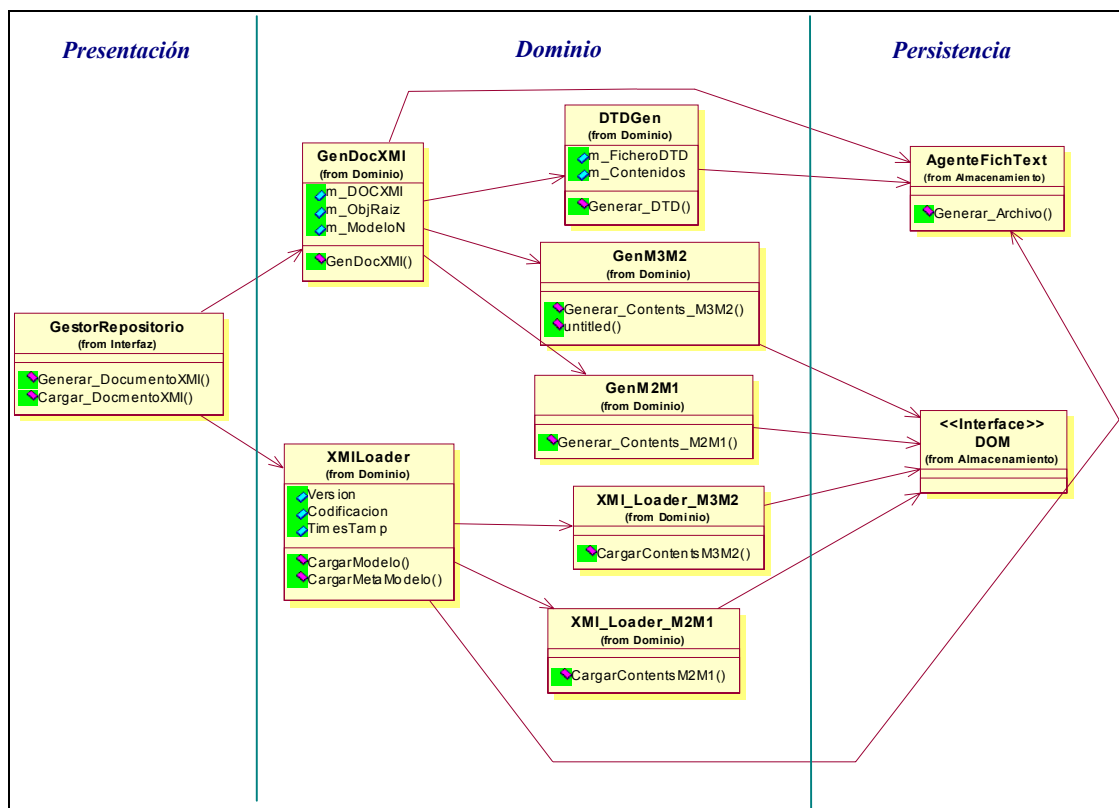


Figura 7-7. Arquitectura de tres capas de RepManager.

Un aspecto importante a considerar es que el gestor de repositorio tiene como objetivo fundamental proporcionar los servicios de almacenamiento persistente a las herramientas de metamodelado MOF, como METAMOD u otras externas a MANTIS. Por lo tanto, desde la

perspectiva de dichas herramientas, RepManager formará parte de la capa de almacenamiento. Igualmente, en la capa de dominio debería incluirse el sub-componente “Implementación MOF” que actuaría de puente de comunicación entre la herramienta y el gestor del repositorio (ver apartado 7.3.2).

7.2.2.3. Componente “Implementación MOF”.

Al definir la arquitectura de RepManager se comprobó que era necesario desarrollar un sub-componente que permitiera gestionar elementos de metamodelos basados en MOF. Puesto que esta funcionalidad era también útil para otras herramientas del Entorno MANTIS como METAMOD, y también podría ser útil en algún momento para otras herramientas externas, se decidió que fuera un componente independiente al gestor del repositorio.

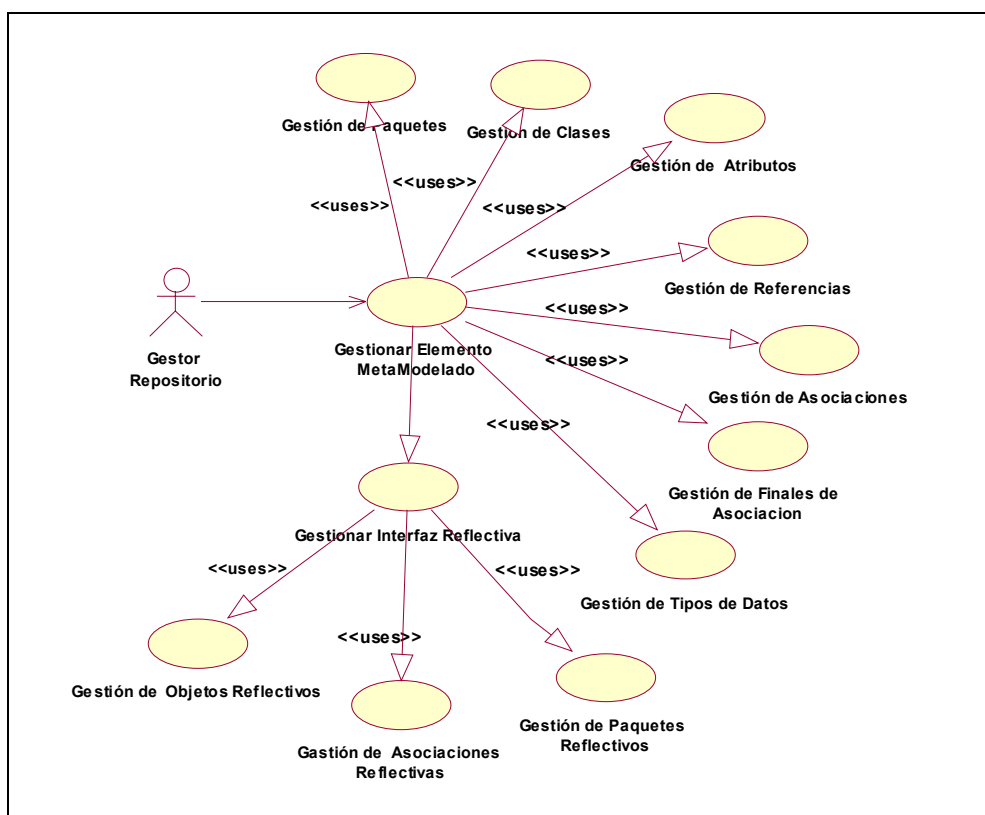


Figura 7-8. Diagrama UML de casos de uso del sub-componente “Implementación MOF”.

La Figura 7-8 muestra el diagrama de casos de uso de este componente. Como se puede observar, la funcionalidad aportada se basa en la gestión de elementos de metamodelado. Los elementos de metamodelado básicos del modelo MOF son los paquetes, las clases, las asociaciones, los atributos, las referencias, los finales de asociación, los tipos de datos y los elementos de la interfaz reflectiva (anexo C). Los elementos de la interfaz reflectiva permiten la instanciación de los meta-elementos para la construcción de modelos. La gestión de cada elemento de metamodelado consiste básicamente en la creación y lectura de la información contenida en el mismo. Este componente es invocado tanto desde los servicios de importación y exportación del gestor de repositorios como desde las herramientas de metamodelado.

7.2.3. Gestión de la Base de Conocimiento: KM-MANTIS.

En el capítulo 6 se ha justificado el interés de utilizar gestión de conocimiento para la mejora del PMS y, en consecuencia, su utilidad para los fines del Entorno MANTIS. También se comentaron las características generales y los fundamentos de la opción elegida. A continuación, se amplía dicha propuesta con la descripción de la arquitectura software elegida para el prototipo (Vizcaíno et al, 2002), si bien éste está en fase de diseño y no ha sido implementado todavía.

7.2.3.1. Arquitectura del Sistema.

KM-MANTIS está formado por varias comunidades de agentes software que manejan diferentes tipos de conocimiento. También existe un repositorio global compartido por las comunidades. Este repositorio global y todos los repositorios parciales representan conocimiento de dos formas: datos o hechos (almacenados como documentos XML) y metadatos o reglas (almacenados como documentos XMI). Las ventajas de usar los estándares XML y XMI ya han sido comentadas en el apartado 7.2.2. Además, aunque nos referimos a diversos repositorios, se trata de una división virtual arquitectural y no física ya que, en realidad, todos ellos se almacenan físicamente juntos en una única base de conocimientos integrada. Dicha base de conocimientos está formada por todos los documentos XML y XMI de los diversos repositorios. Para su gestión se han analizado diferentes alternativas *middleware* y se ha optado por un gestor de datos XML puro, al estilo de Tamino de Software AG, en vez de un gestor de bases de datos objeto-relacional con interfaz para XML (ORACLE 9, Microsoft SQL Server 2002, etc.). La razón de esta decisión es que se necesita almacenar miles de documentos XML, gestionarlos eficientemente y realizar potentes y complejas consultas entre todos ellos para extraer el conocimiento; mientras que las traslaciones XML \leftrightarrow tablas y viceversa, características de la segunda opción, reducen mucho la eficiencia.

El diseño actual de KM-MANTIS está orientado a utilizar las técnicas de gestión de conocimiento para la mejora del proceso de mantenimiento (ver capítulo 8). Para implementar las capacidades de almacenamiento y recuperación de conocimiento se está ampliando el uso que el Entorno MANTIS hace del formalismo REFSENO (ver capítulo 5 y anexo E) para aprovechar los aspectos de implementación de ontologías de dicho formalismo no aprovechados en la versión actual de MANTIS. En concreto, en la base de conocimientos se incluyen todas las características comentadas en las ontologías del capítulo 5 y, adicionalmente, los atributos necesarios para su implementación y para representar instancias: tipos de datos, obligatoriedad de un atributo, valores por defecto, valores inferidos en atributos derivados y pesos estándares. Estos últimos, junto con las “funciones de similaridad entre instancias” (Tautz y Von Wangenheim, 1998), son un mecanismo útil para las consultas complejas necesarias para la recuperación de conocimiento.

Por otro lado, para la implementación de los agentes se ha elegido la plataforma “*FIPA Agent Software Integration Specification*” (FIPA, 2000). Esto permite aprovechar todas las facilidades de definición de roles de agentes, modelos de colaboración, biblioteca de técnicas de razonamiento, etc., que dicha plataforma incorpora. Tal como postula dicha plataforma, el lenguaje basado en XML elegido para la comunicación entre los agentes es el “*Agent Communication Language*” o ACL (FIPA, 2002). Actualmente, de cara a la implementación de KM-MANTIS, se están estudiando dos entornos de desarrollo de agentes basados en FIPA:

- “*Java Agent Development framework*” (<http://sharon.cselt.it/projects/jade>), y

- “ZEUS agent system” (<http://www.cs.iastate.edu/~baojie/acad/current/zeus/zeus.htm>).

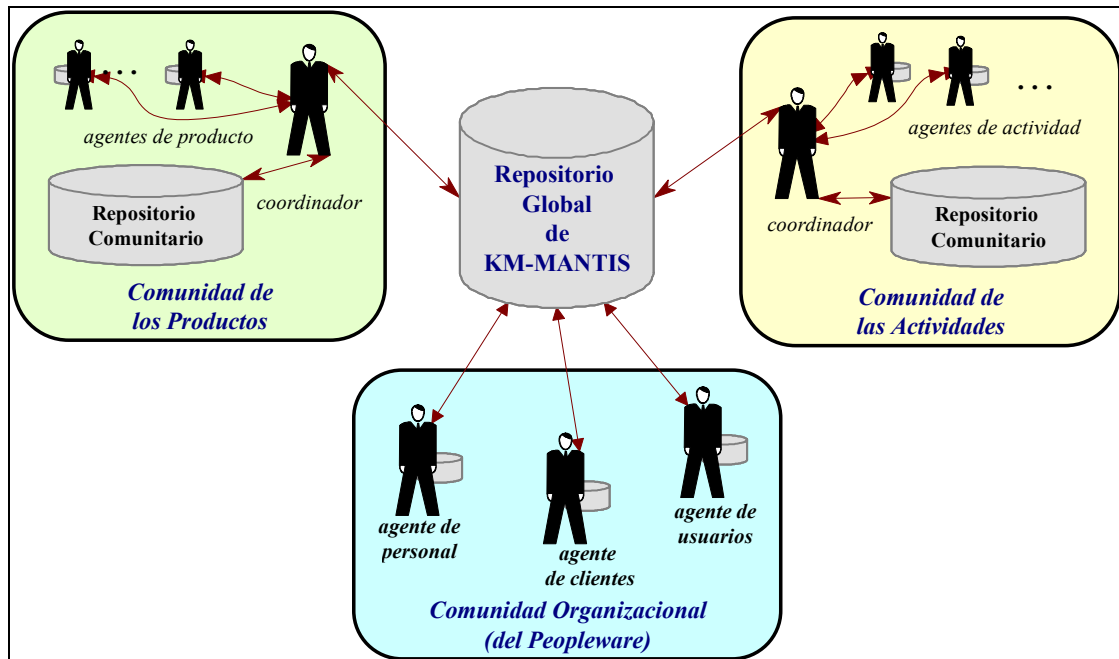


Figura 7-9. Arquitectura de agentes de KM-MANTIS.

7.2.3.1.1. Comunidades de Agentes.

Existe una comunidad por cada uno de los tres grandes tipos diferentes de información detectados en las ontologías presentadas en el capítulo 5 (ver Figura 7-9): los productos que son mantenidos, las actividades realizadas para mantenerlos, y el *peopleware*, es decir, las personas y organizaciones involucradas. Puesto que los procesos definen la manera en que los métodos y herramientas deben ser aplicados para realizar las actividades de mantenimiento y cuales son las habilidades necesarias para llevarlas a cabo (Kitchenham et al, 1999), no existe una comunidad para los procesos porque dicho conocimiento está repartido entre las comunidades de las actividades y del *peopleware*. Estas son las características de los agentes de cada comunidad:

a) Comunidad de los Productos:

Cada producto software tiene sus propios rasgos y sigue una evolución específica, es decir, no sólo los hechos, sino también las reglas que representan conocimiento, son específicos de cada producto software. Por esta razón, en la comunidad de los productos existe un agente diferente para cada producto software sometido a mantenimiento. Así, los agentes de esta comunidad tienen la información sobre los requisitos iniciales, los cambios realizados al producto o las métricas que evalúan características relacionadas con su mantenibilidad. En suma, estos agentes supervisan la evolución de cada producto para tener, en cada momento, la información actualizada sobre él. Para ello, cada vez que un agente detecta que cierta información relacionada con “su producto” está siendo añadida o modificada, el agente se activa y analiza dicha información, la compara con la existente para descubrir inconsistencias, verifica las diferencias y almacena nueva información generada por él mismo (reglas en forma de metadatos) para tener actualizada la base de conocimientos.

Existe conocimiento que no es específico de un producto software en particular sino de todos ellos de forma global. Un ejemplo de este tipo es el catálogo de clases de productos software (componente ejecutable, módulo de código fuente, documento de requisitos, etc.) que el sistema maneja. Todo este conocimiento aprendido es enviado por los agentes de producto a un “agente coordinador” que lo almacena en un repositorio comunitario para poder ser compartido con todos los agentes de producto. El agente coordinador dispone de un catálogo de técnicas de razonamiento que le permiten inferir nuevo conocimiento. Un ejemplo es el siguiente: cuando un agente “descubre” que siempre que su producto tiene una petición de modificación etiquetada como “A” se produce, al poco tiempo, otra posterior etiquetada como “B”, comunica esta regla al coordinador. Éste lo pone en el repositorio comunitario, es decir, lo comparte con toda la comunidad de agentes de producto y, opcionalmente, genera nuevo conocimiento “deduciendo” que, por ejemplo, después de una petición de modificación “A” seguida de otra “B” siempre se produce otra “C” si el producto software es de la categoría “X”.

Además del conocimiento propio de los productos, esta comunidad también registra conocimiento necesario para intercambiar información con las otras comunidades. Así, cada agente “sabe” cuales son las actividades llevadas a cabo para mantener su producto y quienes han sido las personas involucradas ²⁸.

Por último, los agentes de producto también pueden utilizar el repositorio general de KM-MANTIS donde se almacena el conocimiento que no se refiere a productos, y tampoco a actividades, de forma exclusiva.

b) Comunidad de las Actividades:

Esta comunidad tiene una estructura similar a la de la comunidad de los productos, de forma que cada agente tiene su base de datos local y además existe un repositorio comunitario gestionado por un agente coordinador con las mismas funciones que en el caso anterior.

Cada agente de actividad está encargado de gestionar el conocimiento relacionado con una actividad, por ejemplo, cuales son los métodos, técnicas y guías que se pueden utilizar y cuales los recursos necesarios. Los agentes de actividad también pueden generar nuevo conocimiento mediante aprendizaje como, por ejemplo, cuales son los métodos que producen mejores resultados en cada actividad. Como en la comunidad de los productos, los agentes también tienen conocimiento que referencia a otras comunidades: para qué productos se realiza la actividad y cuales son las personas involucradas.

En el repositorio comunitario se almacena el conocimiento importante para toda la comunidad, es decir, de interés para todas las actividades. Un ejemplo es la taxonomía de tipos de actividades definida en el proyecto de mantenimiento en cuestión.

c) Comunidad del Peopleware:

Esta comunidad es la encargada del conocimiento referido a las personas y organizaciones involucradas en el mantenimiento. Esta comunidad tiene una estructura diferente a las dos anteriores debido a que sólo incluye tres agentes, uno por cada tipo de organización participante en el proceso de mantenimiento según se definieron en la sub-ontología de los agentes (capítulo 5): mantenedor, cliente y usuario.

²⁸ Los problemas de integridad derivados de la existencia del equivalente a claves ajenas entre unas comunidades y otras (o entre agentes de la misma comunidad), es decir, entre documentos XML diferentes, están resueltos en la versión 4.0 del servidor XML Tamino.

El agente de personal se encarga del conocimiento referido a las personas que llevan a cabo los proyectos de mantenimiento, o sea, que son miembros de la organización mantenedor. Este conocimiento incluye sus datos personales y laborales, las actividades en que han participado y los productos que han mantenido. Por tanto, este agente sabe, además de la situación actual de cada empleado del mantenedor, la información histórica de ellos. Además, este agente puede generar nuevo conocimiento con la información anterior. Por ejemplo, puede obtener valores estadísticos que indiquen el tiempo medio que un empleado dedicó a realizar una determinada tarea en los proyectos de los últimos 5 años.

El agente de clientes gestiona la información de las organizaciones clientes del mantenedor, sus requisitos (incluso los requisitos iniciales si estuvieran disponibles) y sus demandas. El agente de cliente también intenta obtener nuevo conocimiento como, por ejemplo, prever cambios futuros en los requisitos a partir de los requisitos actuales o estimar los costes de las peticiones de modificación que hace un cliente.

El agente de usuarios está encargado de conocer las necesidades de los usuarios de cada producto, sus entornos de utilización de los productos y sus quejas y comentarios sobre dichos productos. También pueden generar nuevo conocimiento de esta información, por ejemplo, identificando categorías de usuarios y relacionando estas con demandas de nuevas funcionalidades en los productos mantenidos.

Cada uno de estos tres agentes tiene su base de datos local y no existe un repositorio comunitario porque el repositorio global de KM-MANTIS desempeña dicha función.

7.2.3.1.2. Colaboración entre Agentes.

La arquitectura software propuesta implica cierta redundancia de información lo cual podría originar problemas de control de consistencia aunque tiene la ventaja de que los agentes tienen toda la información que necesitan para actuar de forma autónoma. Además, en el diseño actual del prototipo de KM-MANTIS se delega la gestión de dicho control al servidor XML que hace de *middleware* para gestionar físicamente la base de conocimientos, es decir, el conjunto de archivos XML correspondientes.

La arquitectura permite dos niveles de colaboración entre agentes. El primer nivel se refiere a agentes pertenecientes a la misma comunidad. Por ejemplo, cuando el agente de usuarios (comunidad del *peopleware*) pregunta al agente de personal por el empleado que realizó una actividad específica. El caso más frecuente de cooperación de este nivel se produce entre un agente de producto o de actividad y su correspondiente coordinador. El segundo nivel de colaboración se produce entre agentes de diferentes comunidades. En este caso pueden ocurrir tres tipos de comunicación:

- Entre dos agentes no coordinadores. Ejemplo de este tipo sería que un agente de actividad pida al agente de personal información sobre el miembro que lleva a cabo la actividad correspondiente.
- Entre dos agentes coordinadores, es decir, entre el coordinador de los productos y el de las actividades. Por ejemplo, cuando se consultan mutuamente para realizar estimaciones.
- Entre un agente coordinador y otro que no lo es, ambos pertenecientes a comunidades diferentes. Un ejemplo de este tipo ocurre cuando un agente de producto pregunta al coordinador de actividades el número de actividades llevadas a cabo durante el último trimestre en el proyecto X.

Estas clases de colaboración y comunicación permiten que no toda la información tenga que ser duplicada. Además, el conocimiento de cada agente o comunidad puede ser compartido y utilizado por los demás agentes y comunidades, es decir, esta arquitectura permite la reutilización del conocimiento tanto a alto nivel (por los usuarios del sistema) como a bajo nivel (por los agentes software).

7.2.3.1.3. Responsabilidades de los Agentes.

Aunque cada agente desempeña un rol determinado por la comunidad a la que pertenece y por la clase de agente que es, todos los agentes incorporan, de forma general, las siguientes responsabilidades:

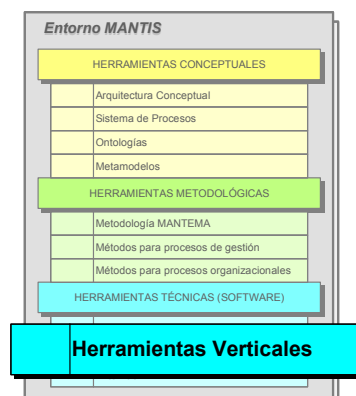
- Comparar la nueva información que reciben con la que ya tienen registrada para detectar inconsistencias y, en su caso, comunicar con los agentes que tienen información relacionada para descubrir cuándo, dónde y porqué se produjo la inconsistencia.
- Informar a los demás agentes afectados de los cambios producidos. Por ejemplo, si el agente del personal registra que la persona X es sustituida por la Y, deberá notificar el cambio a todos los agentes de producto y de actividad en que estuviera participando la persona X.
- Predecir nuevas demandas de los clientes o usuarios. Esta responsabilidad es llevada a cabo principalmente por los agentes de producto ya que productos software similares requieren un mantenimiento similar. La idea base de esta predicción es que lo que una organización ha realizado hasta ahora es útil para predecir lo que hará en el futuro (Gupta y Govindarajan, 2000). Esta responsabilidad es muy importante porque múltiples estudios demuestran que el mantenimiento perfectivo es el más requerido (Piattini et al, 2000). Como dicen Bennet y Rajlich (2000), si los cambios pueden ser anticipados, entonces pueden ser realizados mediante alguna forma de parametrización y, en consecuencia, se podrían reducir los costes y esfuerzos del mantenimiento.
- Utilizar la información histórica para predecir posibles errores. Esta es una de las ventajas de la gestión del conocimiento (Nebus, 2001). En KM-MANTIS se concreta en que cuando el coordinador de actividades detecta un error durante el desarrollo de una actividad, dicho error es comunicado a todos los agentes de actividades para intentar evitar que se repita.
- Sugerir soluciones a los problemas. Zell (2001) postula que las organizaciones están obligadas a reinventar prácticas anteriores debido a una compartición limitada de conocimiento. Para evitarlo, en KM-MANTIS se encarga a los agentes coordinadores de productos y actividades el registro de las prácticas que han dado buenos resultados anteriormente.
- Ayudar a tomar decisiones. Este es uno de los principales objetivos de los sistemas de gestión del conocimiento y, por supuesto, de KM-MANTIS. Cuando el conocimiento mejora es más fácil identificar los problemas, desarrollar soluciones alternativas y seleccionar la mejor solución (Gnyawali et al, 1997). Un ejemplo de esta responsabilidad es que KM-MANTIS (y en especial la comunidad de las actividades) puede ayudar a decidir qué actividades externalizar. Otro ejemplo es aprovechar el conocimiento que el agente de personal tiene sobre todos los empleados (habilidades, métricas de productividad, etc.) para recomendar a uno de ellos como el idóneo para encargarse de una actividad.
- Ayudar a mejorar las estimaciones de los costes o el esfuerzo de las peticiones de mantenimiento. Esto será posible gracias a que con KM-MANTIS se podrá disponer de mucha más información histórica y más elaborada.

Para poder llevar a cabo las responsabilidades anteriores, los agentes utilizan información que tiene tres orígenes diferentes:

- a) Recibida explícitamente, es decir, suministrada directamente por los actores humanos usuarios del sistema. Un ejemplo de este tipo es el nombre de la persona que realiza una actividad.
- b) Generada por los propios agentes. Para ello los agentes utilizan una serie de técnicas de razonamiento automático. Por ejemplo, para estimar en base a casos anteriores se pueden utilizar técnicas de detección de analogías (Carr y Wagner, 2002). Una ventaja de utilizar la arquitectura multi-agente propuesta es que cada agente puede tener sus propias técnicas de razonamiento en función de sus responsabilidades: algoritmos genéticos para tareas de optimización, o redes neuronales para detectar asociaciones o patrones complejos en los datos. El tipo de razonamiento está oculto al sistema, es decir, cada agente actúa con principio de caja negra ofreciendo unas capacidades de razonamiento al sistema sin saber cómo están implementadas.
- c) Compartida por otros agentes tal como ya ha sido comentada previamente.

7.3. Herramientas Verticales.

Estas herramientas han sido diseñadas para automatizar algunas de las actividades pertenecientes a los procesos de gestión y de organización definidos en el sistema de procesos de MANTIS (capítulo 5). Más concretamente, estos prototipos han sido construidos para dar soporte automático a las actividades, métodos, técnicas o guías presentados en el capítulo 6 (Ruiz y Piattini, 2001). A continuación, se presentan los siguientes prototipos: MANTOOL, gestor de peticiones de modificación; METAMOD, herramienta CASE para la gestión de modelos del nivel M1 y meta-modelos del nivel M2 y para establecer las correspondencias M2-M1 y M3-M2 respectivas; MANTICA, para calcular métricas de mantenibilidad de esquemas de datos; SREM, para la estimación de recursos; CREM, para gestionar los recursos humanos en una cartera de proyectos de mantenimiento; y MAN-Quest para evaluar la madurez de servicios de mantenimiento de software.



7.3.1. Gestión de Peticiones de Modificación: MANTOOL.

La primera versión de MANTOOL fue desarrollada para gestionar peticiones de modificación (PM) en base al modelo de procesos definido en la metodología MANTEMA (anexo B). Esta versión se realizó antes de trabajar en la definición del Entorno MANTIS y, por tanto, era la típica herramienta diseñada e implementada de forma completamente independiente sin tener en cuenta ninguna consideración de integración con otras herramientas o con un entorno integrado. En cambio, la segunda versión de MANTOOL ha sido diseñada para integrarse en el Entorno MANTIS (Polo et al, 2001d). Esto ha obligado a realizar un proceso de reingeniería para satisfacer los nuevos requisitos técnicos (ver capítulo 5); si bien, desde el punto de vista del usuario, la funcionalidad prácticamente no ha cambiado salvo en un aspecto fundamental: ahora es posible elegir otros modelos de PMS, diferentes al definido por

MANTEMA, entre los modelos del nivel M1 almacenados en el repositorio general de MANTIS. Esto significa que la nueva versión está completamente parametrizada, lo que redundará en una mayor flexibilidad y adaptación a las características de cada organización o proyecto. Esta nueva posibilidad sólo está restringida por el conocimiento del dominio del problema establecido en las ontologías de MANTIS y, en especial, en la sub-ontología de organización del proceso (capítulo 5). Esta limitación es necesaria porque es imprescindible que todos, usuarios y sistema, llamen igual a lo mismo y, en especial, porque los usuarios necesitan conocer claramente la nomenclatura y conceptos que se utilizan en la herramienta.

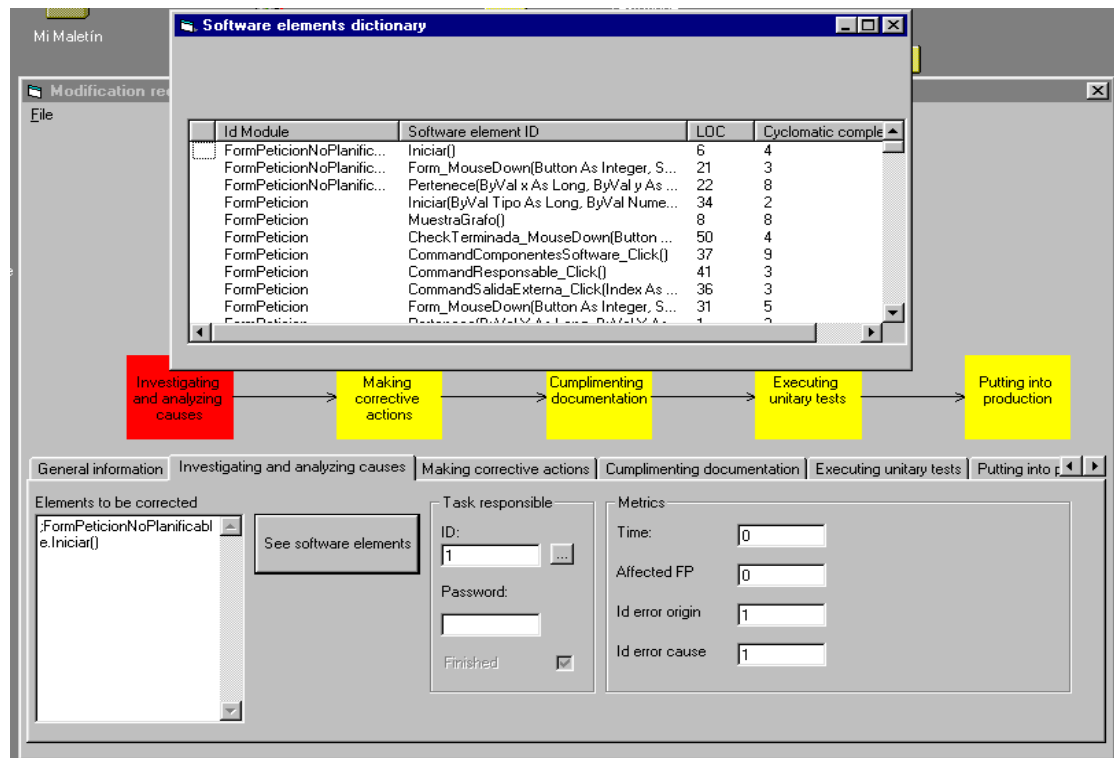


Figura 7-10. MANTOOL: estado actual de una petición de modificación.

MANTOOL contempla la gestión de PM desde su recepción por el responsable del mantenimiento (mantenedor) hasta la conclusión de todas las actividades y tareas para atenderlas (que dependen del modelo de proceso utilizado). Cuando el mantenedor recibe una PM de los usuarios, debe elegir un tipo de mantenimiento (entre los definidos en el modelo de proceso activado) y se muestra una pantalla para introducir la información relevante de la PM. Después de esta acción, se muestra una ventana con el diagrama representando el conjunto de actividades definidas para el tipo de mantenimiento elegido. Las actividades ya completadas en cada momento se muestran con un color más oscuro. Debajo del diagrama de flujo se muestra una pestaña con información general de la PM y una pestaña más por cada actividad o tarea definidas. En la Figura 7-10 se muestra la pantalla para una PM de mantenimiento correctivo urgente que ya tiene completada la primera de sus 5 actividades. En el ejemplo mostrado también se observan, en una ventana adicional, todos los elementos software que se ha detectado, durante la primera actividad de investigación y análisis de las causas, que deben ser cambiados.

Los formularios y pantallas están parametrizados en función del modelo de procesos predefinido. Así por ejemplo, los documentos a rellenar deben ser definidos como procedimientos de tipo guía en el modelo de procesos y los documentos XMI que definen la

estructura de dichos documentos se deben almacenar en el repositorio de MANTIS. En la Figura 7-11 se muestra un ejemplo del formulario para la entrada de acciones correctivas.

DOC7-Executed corrective actions

File

Id DOC7: 20

Modification request (DOC6 Id): 71

Responsible: Francisco Ruiz

Telephone: 212578

Programming language: Visual Basic

Free text:

A boolean variable has been added to routine "CargaDatos", whose value is TRUE when the window is closed. The TRUE value is also a condition to exit from the loop of CargaDatos.

Exit Save

Figura 7-11. Ejemplo de formulario en MANTOOL: entrada de acciones correctivas.

7.3.1.1. Informes.

Toda la información capturada por MANTOOL es almacenada en el repositorio de MANTIS en forma de documentos XML (datos) o XMI (metadatos: modelos de procesos, estructura de documentos, etc.). Aprovechando dicha información, en la versión actual de MANTOOL se han incorporado los siguientes tipos de informes (Polo et al, 2001d):

- De aplicación: Muestran un resumen de los diferentes datos almacenados para cada aplicación de manera global o por tipos de mantenimiento; por ejemplo: tiempo dedicado a mantenimiento, número de líneas de código añadidas, número de errores detectados durante las pruebas, etc.
- De tipo de mantenimiento: Muestran en forma de tabla la colección de datos de las PM de un determinado tipo de mantenimiento. Estas tablas pueden ser exportadas, a ASCII o Excel, para su reutilización posterior.
- De dedicación de personal: Muestran el tiempo dedicado a cada actividad por cada persona o por todo el equipo de mantenimiento de forma global.
- De desviaciones: Permiten observar las desviaciones producidas entre el tiempo estimado para realizar cada PM y el realmente dedicado.

- e) De tendencia: En el caso de que en el modelo de procesos se hayan asociado métricas a los artefactos de tipo código fuente (módulos, rutinas, procedimientos, etc.), estos informes permiten ver la evolución de las diversas métricas para los diferentes artefactos a lo largo del tiempo. Por ejemplo, en la Figura 7-12 se muestra la evolución de 6 métricas (parámetros de entrada, parámetros de salida, líneas de código, líneas de comentario, complejidad ciclomática y días transcurridos desde el último cambio) para el módulo “ActualizarAplicación”.

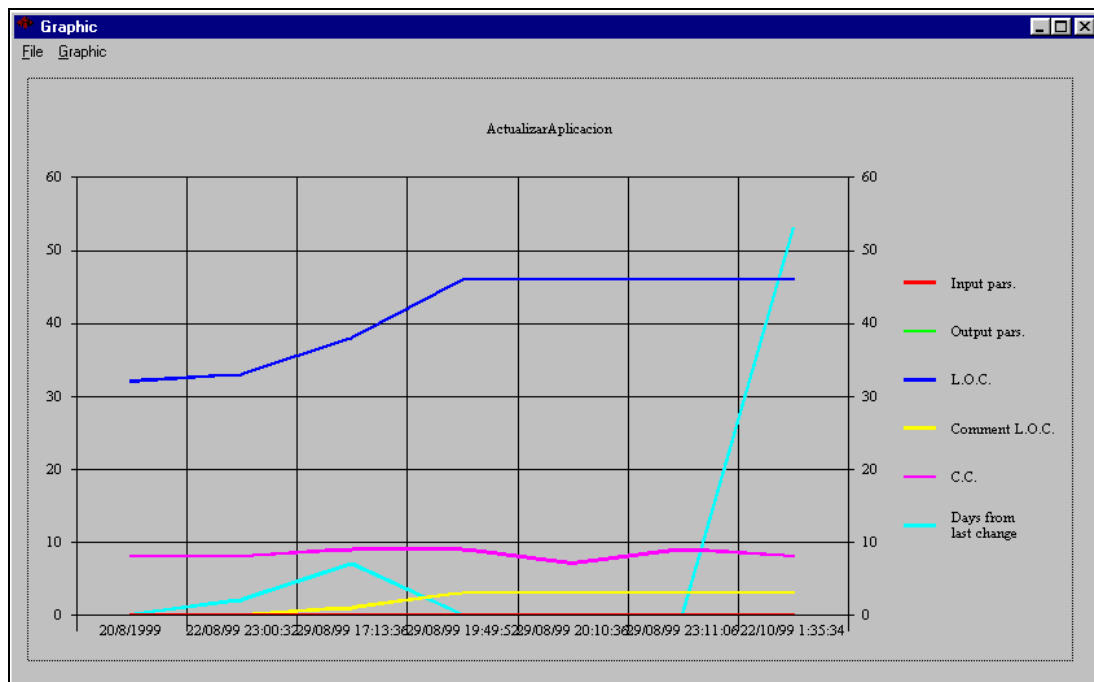


Figura 7-12. MANTOOL: ejemplo de informe de tendencia.

7.3.2. Administración de Modelos y Meta-Modelos: METAMOD.

METAMOD no es una herramienta CASE para el diseño de modelos de proceso o de cualquier otro dominio de aplicación (Ruiz et al, 2001d). Para ello ya existen muy buenas y completas herramientas que incluyen avanzados interfaces gráficos. La razón de desarrollar METAMOD fue cubrir una necesidad, detectada durante el desarrollo del Entorno MANTIS, que no estaba cubierta por ninguna de las herramientas conocidas: poder definir y visualizar las correspondencias entre los elementos de modelos y metamodelos, basados en MOF/XMI, a diferentes niveles de abstracción. Por esta razón, no se intentó incorporar un entorno gráfico de diseño que hubiera resultado muy costoso de desarrollar y se prefirió utilizar una sencilla metáfora visual “jerárquica”. Además, esta metáfora se adapta perfectamente a las características de XMI, el lenguaje de metamodelado basado en MOF utilizado en el repositorio de MANTIS (ver apartado 7.2.2). En el anexo H se puede consultar el manual de usuario de esta herramienta.

Dado que METAMOD es una herramienta para administrar los modelos y metamodelos incluidos en el repositorio de MANTIS, no incluye la funcionalidad de gestionar las correspondencias M1-M0 entre un modelo y los datos de proyectos concretos. Por tanto, con METAMOD es posible administrar meta-metamodelos (es decir, el modelo MOF del nivel M3),

metamodelos del nivel M2, modelos del nivel M1, y sus respectivas correspondencias M3-M2 y M2-M1 (Márquez et al, 2001a; García et al, 2001a).

METAMOD se ha diseñado para satisfacer los siguientes requisitos técnicos y funcionales, adicionales a los derivados de su integración en el Entorno MANTIS:

- Ser una aplicación Windows que utiliza los servicios de cargar y exportar modelos y metamodelos de RepManager. Para manejar en memoria los modelos y metamodelos MOF utilizará el componente “Implementación MOF”, ya comentado en el apartado 7.2.2.3.
- Crear y editar modelos y metamodelos basados en el estándar MOF, utilizando los constructores que éste proporciona (paquetes, clases y asociaciones, principalmente).
- Disponer de un interfaz visual sencillo para mostrar las correspondencias entre objetos de un nivel y sus instancias del nivel inmediatamente inferior; permitiendo trabajar con varios modelos y metamodelos a la vez.

La implementación del prototipo actual de METAMOD se realizó como proyecto fin de carrera por el estudiante Luis Márquez (Márquez, 2001). En el anexo I se muestran varios ejemplos de documentos generados con METAMOD (y RepManager) correspondientes a:

- a) El paquete básico adaptado del “Metamodelo genérico de proceso software” (anexo G). Se incluyen el archivo XMI (adaptado al DTD del modelo MOF) y el DTD del nivel M2.
- b) El modelo de “actividades para mantenimiento no planificable según la metodología MANTEMA” (anexo G). Se Incluye el archivo XMI adaptado al DTD anterior.

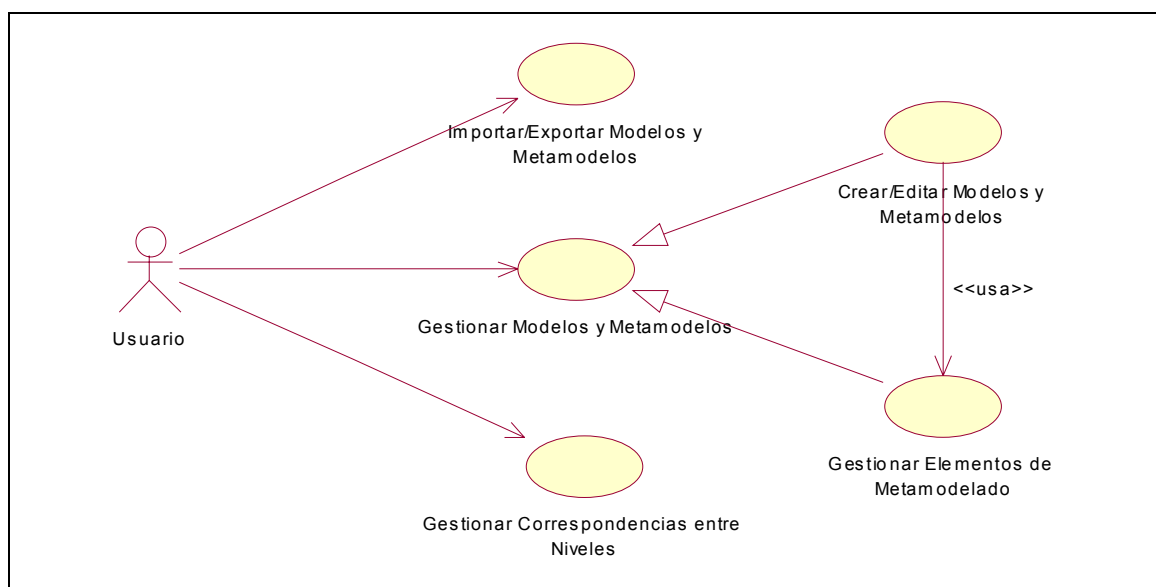


Figura 7-13. METAMOD: Diagrama UML de casos de uso.

7.3.2.1. Casos de Uso.

El único actor que interactúa con la herramienta es un ingeniero software que quiere trabajar con modelos y metamodelos basados en la arquitectura conceptual propuesta por MOF. En la Figura 7-13 se muestra el diagrama UML de casos de uso de esta herramienta. Todos estos casos de uso se describen más en detalle a continuación (Márquez et al, 2001b; 2001c):

- a) Gestionar Correspondencia entre Niveles: Como ya se ha comentado, la principal funcionalidad de la herramienta es poder trabajar a la vez con objetos de determinado nivel (M3, M2 o M1) y con sus correspondencias de nivel superior (meta-objetos) y de nivel inferior (instancias). Esta funcionalidad, que las herramientas CASE conocidas no soportan, le aporta al ingeniero software una nueva capacidad para gestionar metadatos a diferentes niveles de abstracción. Por ejemplo, dado un objeto de un metamodelo de nivel M2, se puede acceder tanto a las instancias de nivel inferior (correspondencias M2-M1), como a su meta-objeto (correspondencias M3-M2).
- b) Gestionar Modelos y Metamodelos: Este caso de uso es el encargado de gestionar un modelo o un metamodelo concreto. Esta gestión contempla tanto su creación como el uso de los diversos constructores de modelado propuestos por MOF. Estas dos opciones se concretan en dos nuevos casos de uso:
 - b.1) Crear y Editar Modelos y Metamodelos: La herramienta debe permitir la creación y edición de modelos y metamodelos, que serán los contenedores de los distintos constructores de MOF. Cabe destacar que el meta-metamodelo de nivel M3 en el Entorno MANTIS siempre es el modelo MOF (ver anexo C), es decir, el DTD de todos los metamodelos del nivel M2 es el DTD de MOF.
 - b.2) Gestión de Elementos de Metamodelado: Este caso de uso recoge toda la funcionalidad que aporta la especificación MOF. Gracias a dicha especificación, METAMOD puede gestionar los modelos y metamodelos necesarios de forma robusta y eficiente; y se pueden manejar los elementos de metamodelado básicos del modelo MOF: paquetes, clases, asociaciones, atributos, referencias, finales de asociación, tipos de datos y elementos de la interfaz reflectiva. Esta funcionalidad es satisfecha por el componente software “Implementación MOF”, que es externo a la aplicación METAMOD (ver apartado 7.2.2.3),.
- c) Importar/Exportar Modelos y Metamodelos: Otra de las funcionalidades básicas de esta herramienta es posibilitar el manejo de modelos y metamodelos creados con otras herramientas diferentes. Esta funcionalidad es soportada por los servicios correspondientes de importación y exportación de RepManager (apartado 7.2.2).

7.3.2.2. Integración con otras Herramientas CASE.

La integración de METAMOD con otras herramientas CASE de diseño se realiza a nivel de datos mediante el uso del lenguaje estándar XMI para intercambio de metadatos. Esto significa que METAMOD puede intercambiar datos y metadatos (modelos y metamodelos) sin problemas con cualquier otra herramienta de modelado que incluya opciones de exportación y/o importación basada en MOF. Como ejemplo cabe destacar a Rational Rose, herramienta para modelado UML que dispone de una utilidad que permite importar y exportar modelos y metamodelos MOF. En la Figura 7-14 se muestra una pantalla de Rational Rose con el diseño de un metamodelo y la opción de menú para su exportación a MOF, es decir, a un documento XMI.

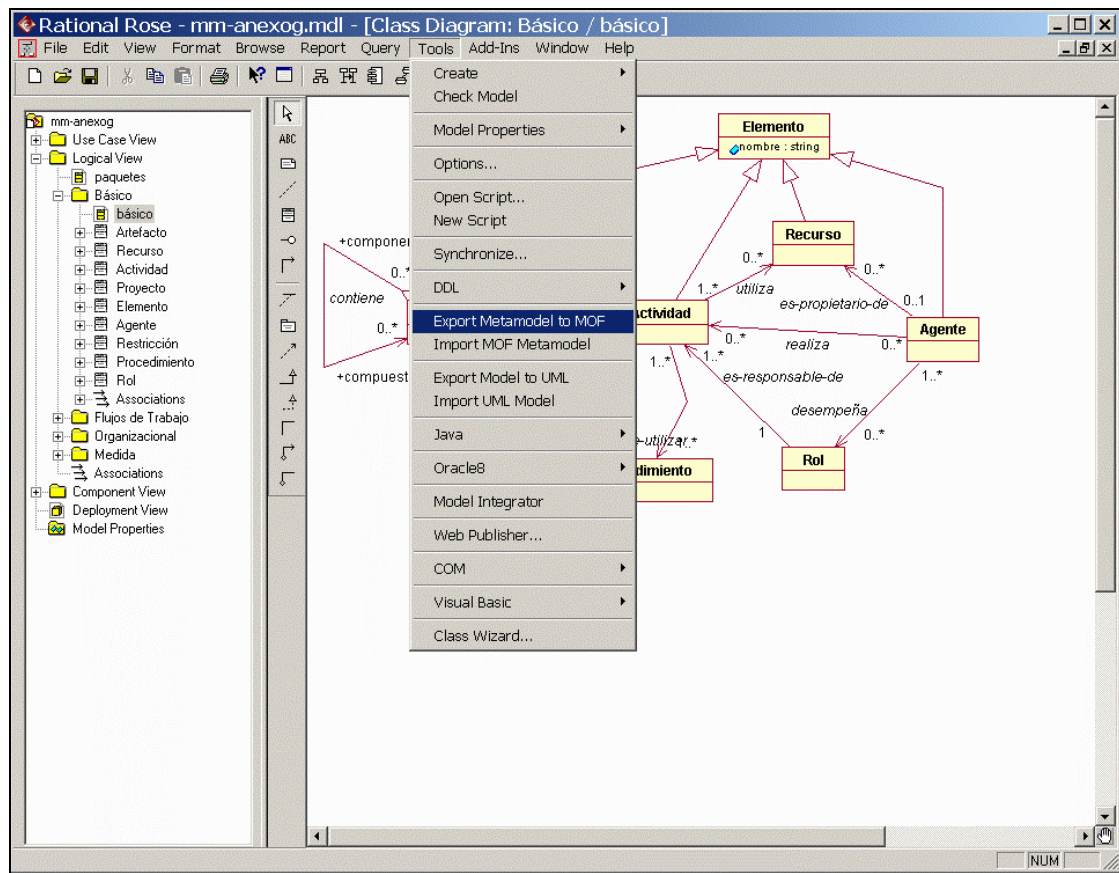


Figura 7-14. Exportación de metamodelos en formato MOF/XML desde Rational Rose.

7.3.2.3. Arquitectura de Tres Capas.

Al igual que las demás herramientas internas de MANTIS, METAMOD también ha sido diseñada con una arquitectura clásica de tres capas:

- Capa de presentación, que está descrita con los casos de uso anteriores y el manual de usuario incluido en el anexo H. Sus principales componentes son los formularios descritos en el manual del citado anexo. La Figura 7-15 es un ejemplo del aspecto visual de esta herramienta. La ventana izquierda muestra todos los elementos de un metamodelo de nivel M2, mientras que la ventana derecha muestra las propiedades de una de las clases de dicho metamodelo.
- Capa de dominio, en la que destaca que, al igual que pasa con RepManager, incluye el componente “Implementación MOF” (apartado 7.2.2.3), que también ha sido desarrollado ex profeso para el Entorno MANTIS.
- Capa de almacenamiento o persistencia, que está formada por el componente RepManager suficientemente comentado en apartados anteriores. La relación funcional entre METAMOD y RepManager se puede observar resumida en la Figura 7-4.

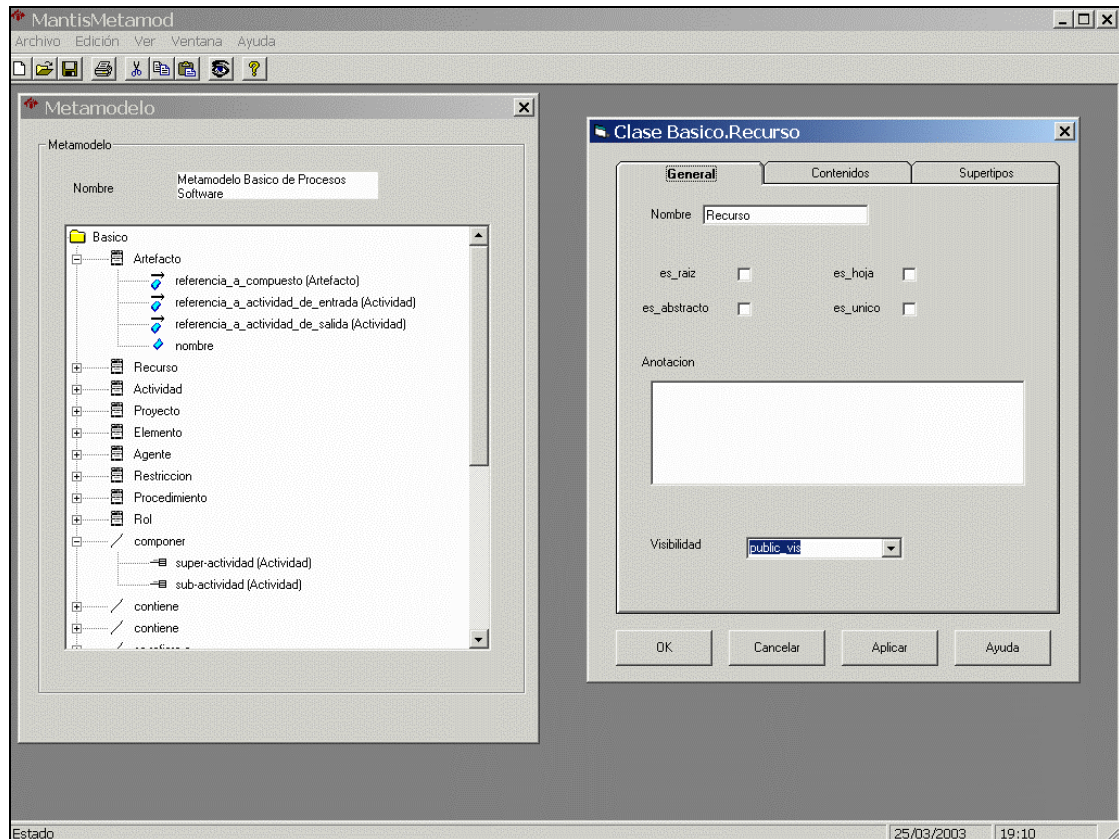


Figura 7-15. METAMOD: Aspecto del interfaz de usuario.

7.3.3. Métricas de Mantenibilidad: MANTICA.

La primera versión de la herramienta MANTICA fue desarrollada con el objetivo, muy concreto, de ayudar a calcular una colección cerrada y predefinida de métricas para la mantenibilidad y calidad de esquemas relacionales y objeto-relacionales de bases de datos. La versión actual de MANTICA, la segunda, ha supuesto la reingeniería de la versión anterior para incorporar los siguientes requisitos (Ruiz et al, 2002b):

- Integración en el Entorno MANTIS, es decir, representación de las métricas y los valores de las mediciones mediante documentos XML en el repositorio de MANTIS, y utilización del marco conceptual para la medición presentado en el capítulo 5.
- Extensión a otros tipos de esquemas de datos: conceptuales entidad/interrelación, de clases UML; etc. Esto implica una mayor parametrización que se puede llevar a cabo gracias a la arquitectura multinivel de MANTIS. En concreto, cada modelo de datos (relacional, UML, entidad/interrelación,) tiene un metamodelo de nivel M2 y cada esquema se representa mediante un modelo del nivel M1 asociado a su correspondiente metamodelo.

La razón de atender a las métricas citadas fue que, siendo la mantenibilidad de los esquemas de datos uno de los factores que influye en el coste posterior del mantenimiento de software (Piattini et al, 2000), no existían herramientas conocidas ni métricas definidas que abordaran este problema. Algunas de las métricas incluidas han sido propuestas por otros miembros del Grupo Alarcos (Calero et al, 2000).

El prototipo actual de MANTICA fue desarrollado como proyecto fin de carrera por el alumno Manuel Ángel Serrano (Serrano, 2000). La funcionalidad que aporta esta herramienta viene determinada por los siguientes casos de uso:

- Importar Esquema:** Incorporar al repositorio esquemas de datos definidos en diversos formatos (SQL estándar en ASCII plano, Access, ORACLE, SQL Server, etc.). Todos estos esquemas se almacenan en forma de documentos XML cuyo DTD es el definido por el correspondiente metamodelo (relacional, UML, etc.). Esto tiene la ventaja de que es necesario un algoritmo de importación diferente para cada uno de los formatos de origen, pero todas las demás partes de la aplicación son comunes.
- Visualizar Esquema:** Observar, mediante una metáfora basada en listas de elementos, los componentes de un esquema.
- Mostrar Información de un Esquema:** Abrir una ventana para ver y editar los datos generales de un esquema: autor, fecha de creación, versión, etc.
- Comparar Esquemas:** Permite elegir dos o más esquemas para mostrar una comparativa, en forma de tabla o de gráfico, de los valores que adoptan en ellos unas determinadas métricas. En la Figura 7-16 se muestra el aspecto de MANTICA con una ventana comparando varios esquemas (modelos en la terminología de MANTIS) para diversas métricas elegidas entre la lista de las disponibles para dichos esquemas.

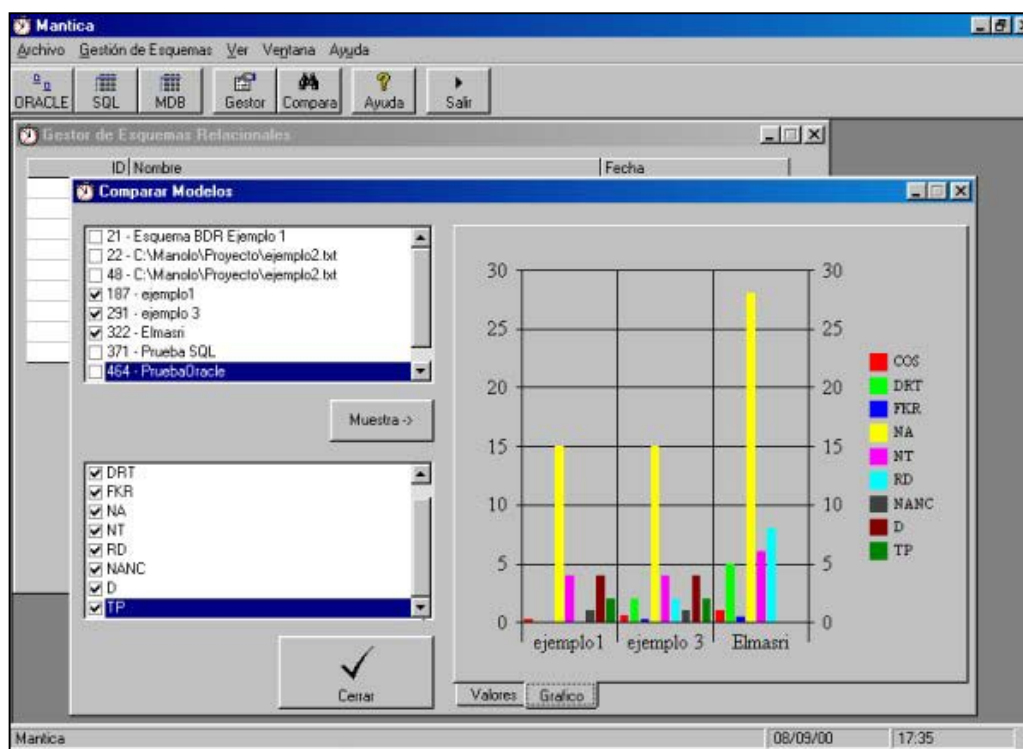


Figura 7-16. MANTICA: Formulario de comparación de modelos.

Conforme se ha ido avanzando en la definición del Entorno MANTIS se ha ido haciendo más patente el papel central que la medición juega en la gestión de proyectos de mantenimiento (o de cualquier otro tipo). Por ello, recientemente se ha tomado la decisión de

añadir al sistema software de MANTIS, presentado al principio de este capítulo, un nuevo componente horizontal encargado de dar soporte globalizado al proceso de medición. El objetivo es, explotando al máximo las posibilidades de generalidad que aporta el marco de trabajo de MANTIS, diseñar y construir una herramienta extensible para la definición, cálculo y visualización de cualquier tipo de métricas, tanto de producto como de proceso (García et al, 2003b). Más detalles de esta nueva línea de trabajo se comentan en el capítulo 8 (proyecto GenMETRIC).

7.3.4. Estimación de Recursos: SREM.

SREM (*Single tool for Resources Estimating in Maintenance*) es una aplicación para realizar los cálculos planteados por el método de estimación de los recursos humanos necesarios para mantenimiento urgente correctivo (es decir, mantenimiento cuyas intervenciones de modificación no pueden tener planificación) presentado en el capítulo 6. Se trata pues de una traslación directa de las fórmulas ya comentadas.

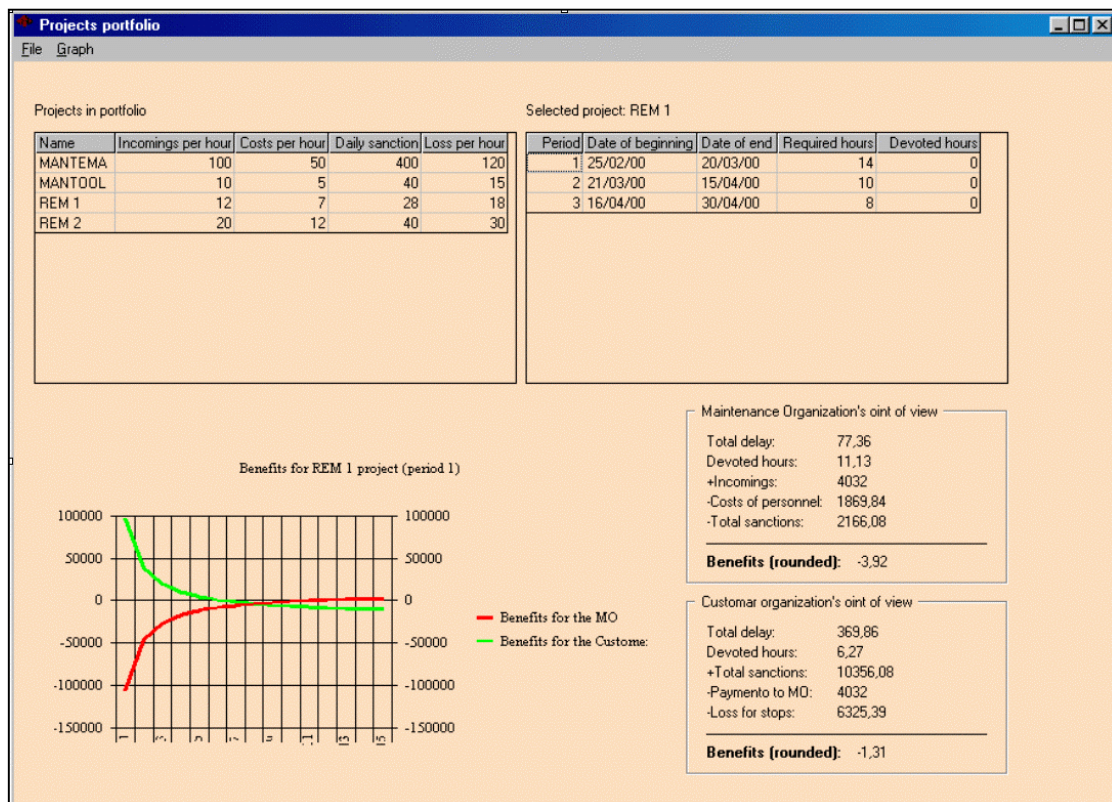


Figura 7-17. SREM: Pantalla principal.

La Figura 7-17 muestra la pantalla principal con los principales resultados que la herramienta aporta. La esquina superior izquierda muestra la cartera de proyectos, es decir, la lista de proyectos en curso. Para cada proyecto se registran los siguientes datos: nombre, precio que el cliente paga al mantenedor por cada hora-persona de recurso contratada (μ_c), coste que para el mantenedor soporta por cada hora-persona (μ_m), penalización que el mantenedor deja de ingresar por cada día de retraso (s), y pérdida que le supone al cliente cada hora sin poder utilizar la aplicación (δ). La tabla mostrada en la esquina superior derecha muestra, para el

proyecto elegido previamente, una división en periodos temporales (también elegidos por el usuario) y, para cada periodo, las fechas de comienzo y finalización, el número de horas acordadas y el número de horas realmente dedicadas (cuando ya es conocido).

En la parte inferior izquierda se muestra un gráfico con la evolución de los beneficios del mantenedor, según el modelo de costes/beneficios comentado en el capítulo anterior. Este análisis coste/beneficios corresponde al periodo y proyecto seleccionados anteriormente por el usuario. Los datos numéricos resumidos se muestran también a la derecha del gráfico.

Además de los cálculos desde el punto de vista del mantenedor, que son los correspondientes a las ecuaciones del capítulo 6, la herramienta también incluye los cálculos equivalentes desde el punto de vista de la organización cliente. El método de cálculo es similar pero teniendo en cuenta que lo que son ingresos para uno son costes para el otro y viceversa. La única variable nueva introducida para poder realizar este segundo análisis costes/beneficios es que los costes del cliente incluyen la nueva variable δ , es decir, la pérdida estimada por no poder utilizar la aplicación durante una hora.

La versión actual de SREM no sirve para el caso de organizaciones que desempeñan a la vez los roles de mantenedor y cliente, es decir, organizaciones que mantienen su propio software.

7.3.5. Gestión de Recursos: CREM.

La herramienta CREM tiene por objetivo ayudar a gestionar los recursos humanos en organizaciones que tienen uno o varios proyectos de mantenimiento. En concreto, se utiliza el método propuesto en el capítulo 6 para decidir la manera más económica de repartir los recursos humanos entre las diversas peticiones de modificación (PM) pendientes en cada momento, buscando minimizar las pérdidas por penalizaciones debidas a retrasos.

Los requisitos funcionales y técnicos del prototipo actual, desarrollado como proyecto fin de carrera por la alumna Maria Dolores Mateos (Mateos, 2001), fueron los siguientes:

- Debe permitir realizar automáticamente los cálculos necesarios para optimizar los beneficios del mantenedor a la hora de decidir el reparto de los recursos humanos, utilizando el modelo de costes/beneficios presentado en el capítulo 6.
- La herramienta deberá servir para el caso de tener uno o varios proyectos de mantenimiento a la vez.
- En cualquier momento se deben poder añadir nuevos proyectos, nuevas PM y nuevos recursos.
- Los datos de los proyectos, peticiones y recursos deben poderse modificar en cualquier momento.
- Se podrán obtener informes de los proyectos, peticiones y recursos de una cartera. También se dispondrá de informes de las asignaciones de recursos y peticiones obtenidas para cada día.
- Los informes se podrán imprimir o exportar a diferentes formatos.
- La asignación optimizada de recursos entre las PM y días deberá ser fácil de comprender por el usuario.
- También se deberán indicar los beneficios de los diversos proyectos.
- Todos los datos manejados por CREM se almacenan en formato XML en el repositorio de MANTIS.

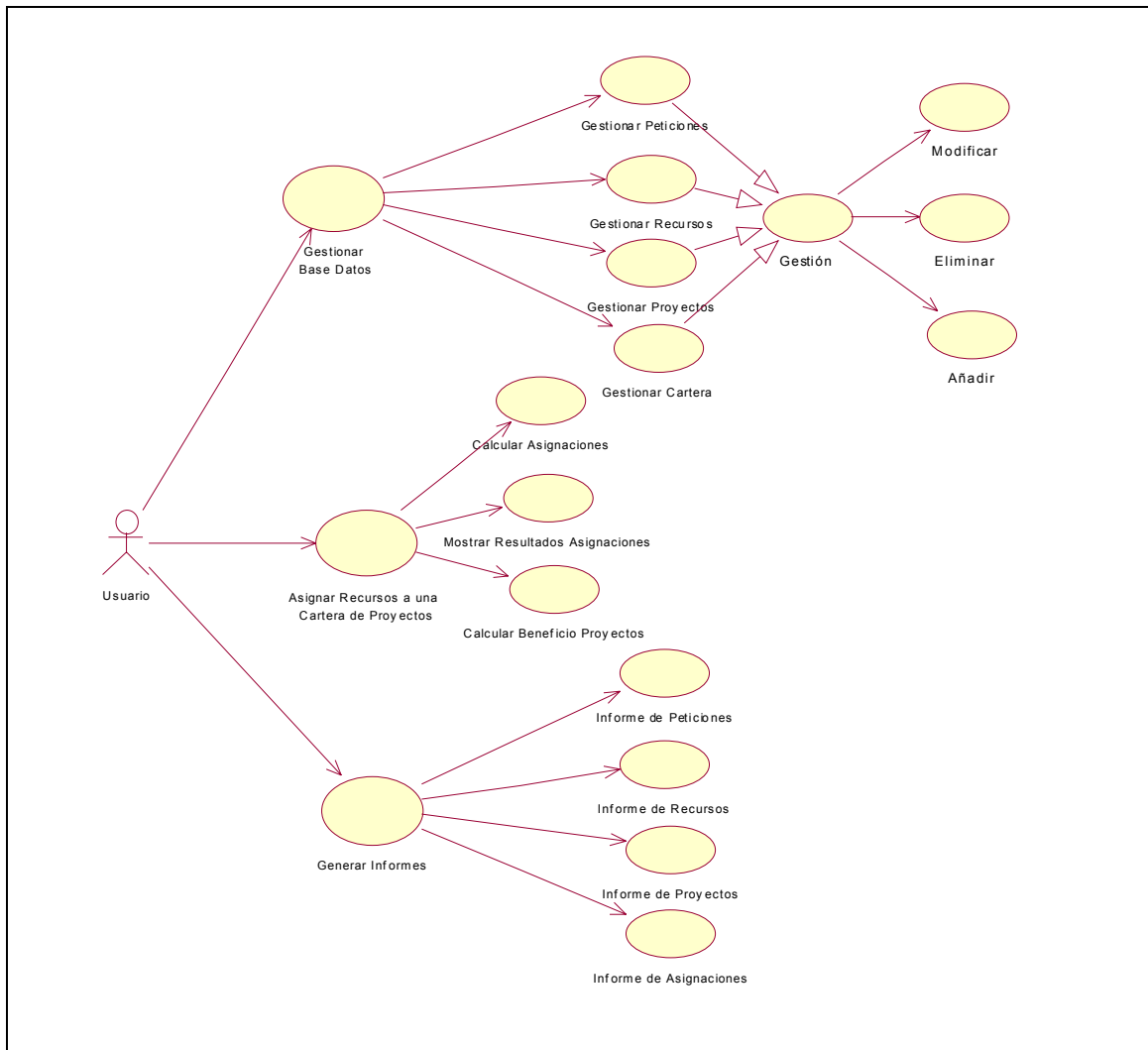


Figura 7-18. CREM: Diagrama UML de casos de uso.

7.3.5.1. Casos de Uso.

Sólo existe un actor a considerar, que es el usuario de la aplicación (normalmente un ingeniero de mantenimiento). Como se puede observar en la Figura 7-18, existen tres casos de uso principales (ver manual de usuario incluido en el anexo H para más información):

- Gestionar Base de Datos*: El usuario podrá añadir, modificar y eliminar los datos que desee de carteras (pueden tenerse almacenadas diferentes carteras de proyectos), recursos, proyectos y peticiones. Este caso de uso está dividido en otros cuatro: “gestionar peticiones”, “gestionar recursos”, “gestionar proyectos” y “gestionar cartera”. A su vez, cada uno de estos casos de uso puede utilizar los casos de uso genéricos “añadir”, “eliminar” y “modificar”, que representan las operaciones posibles que la herramienta permite hacer sobre la base de datos. A título de ejemplo, la Figura 7-19 muestra el diagrama de secuencia del caso de uso “Añadir PM”.
- Asignar Recursos a una Cartera de Proyectos*: El usuario obtendrá la información referente a la distribución de recursos para la cartera de proyectos seleccionada, incluyendo los beneficios de cada uno de los proyectos. Este caso utiliza el caso genérico “gestionar”, que

realmente consta de los tres siguientes: “calcular asignaciones” (ver Figura 7-20), “mostrar resultados asignaciones” y “calcular beneficio proyectos”.

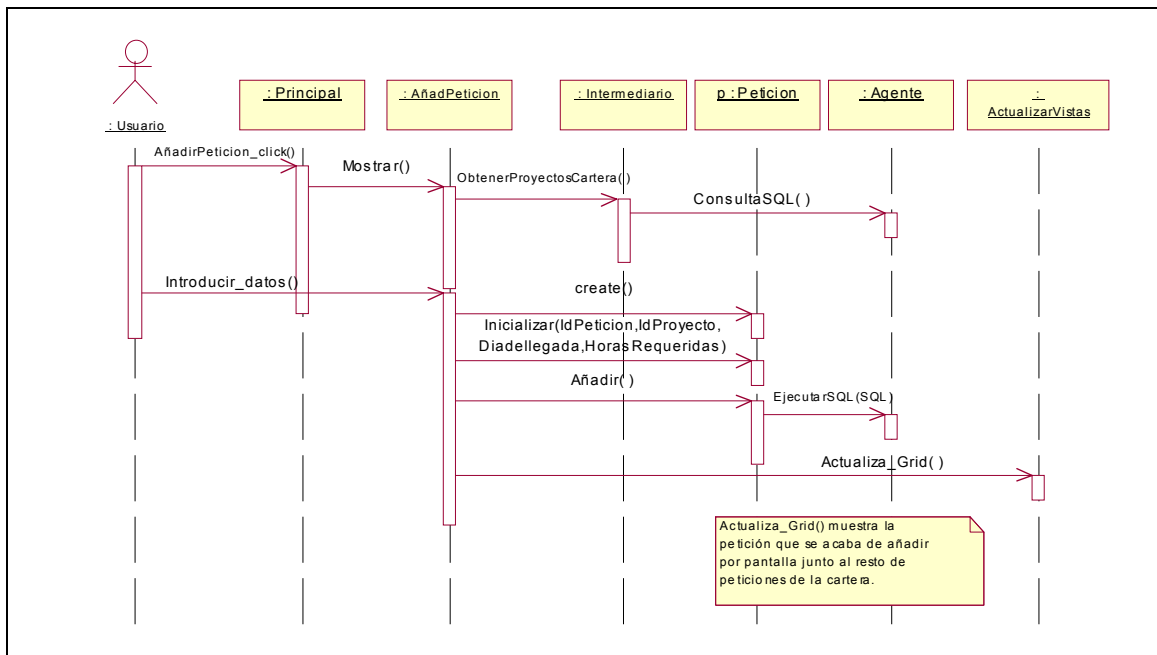


Figura 7-19. CREM: Diagrama de secuencia del caso de uso "añadir PM".

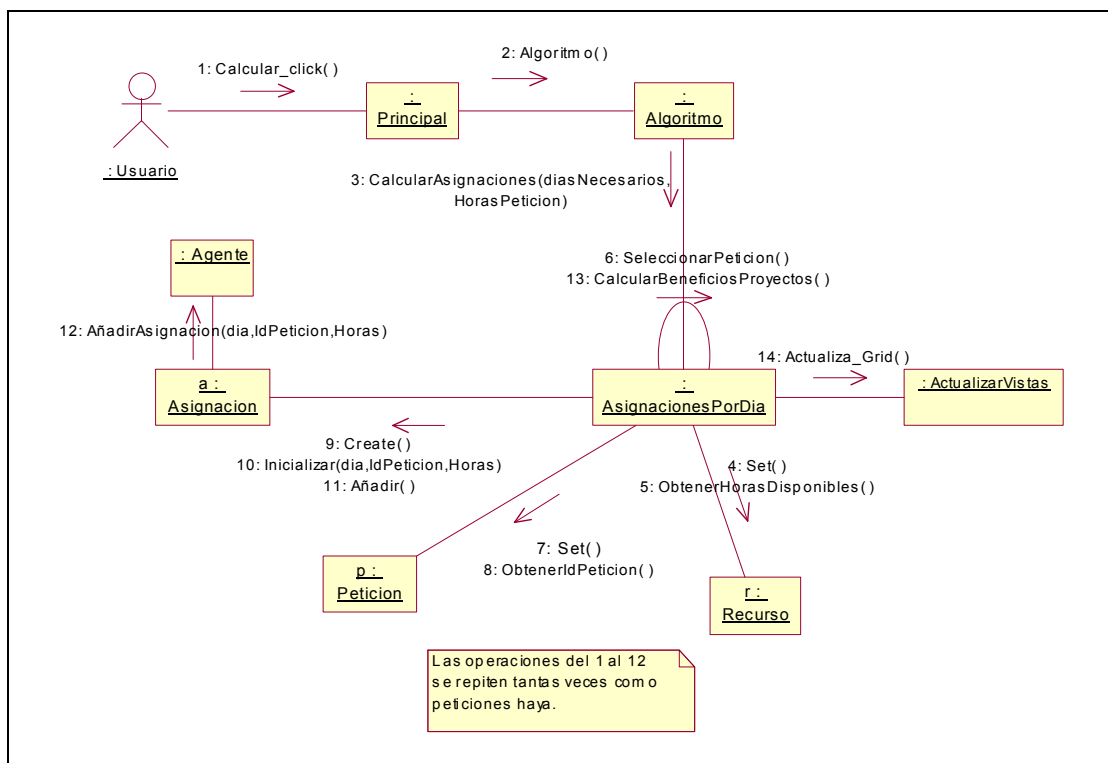


Figura 7-20. CREM: Diagrama de colaboración del caso de uso "calcular asignaciones".

- c) *Generar Informes*: El usuario podrá emitir informes de peticiones, productos, proyectos o asignaciones de los recursos de una cartera. También podrá exportarlos. Este caso de uso también se divide en otros cuatro, que son “informe de peticiones”, “informe de recursos”, “informe de proyectos” e “informe de asignaciones”.

7.3.5.2. Arquitectura de Tres Capas.

La arquitectura de tres capas de CREM, seguida en todas las aplicaciones del Entorno MANTIS, se muestra en la Figura 7-21. Los contenidos de cada capa son:

- La capa de presentación incluye la clase del formulario principal y clases para las demás ventanas de la aplicación: ElegirCartera, EliminarCartera, AñadirCartera, AñadirProyecto, AñadirPetición, AñadirRecursos, AñadirIntervaloRecurso, ModificarRecursos, ModificarPetición, ModificarProyecto, InformeAsignaciones, InformeProyectos, etc.
- La capa de dominio incluye las clases con los algoritmos de cálculo y las correspondientes a los clases de objetos que se almacenan en el repositorio: Cartera, Proyecto, Recurso, Petición, Asignación. Esto último es debido a que se ha utilizado el patrón “una clase una tabla”. También existe una clase “Observador” encargada de comunicar con la capa de almacenamiento.
- La capa de persistencia o almacenamiento sólo contiene la clase “Agente” que se comunica con un interfaz “conexión”. En el prototipo actual de CREM este interfaz es RepManager, el gestor del repositorio de MANTIS.

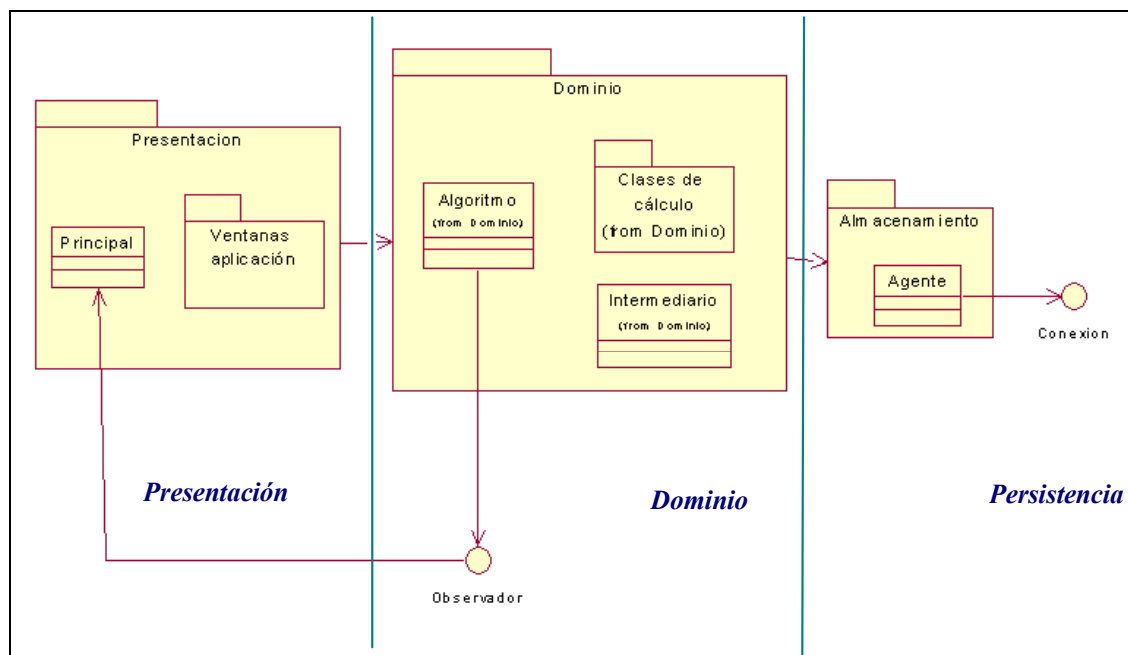


Figura 7-21. CREM: arquitectura de tres capas.

7.3.6. Evaluación de la Madurez: MAN-Quest.

La primera versión de esta herramienta se desarrolló con el único objetivo de facilitar la cumplimentación del cuestionario para evaluar la madurez de servicios de mantenimiento, que se ha presentado en el capítulo 6. Los requisitos establecidos para alcanzar este objetivo fueron los siguientes:

- Poder evaluar servicios de mantenimiento según el cuestionario incluido en el capítulo 6, es decir, determinar los niveles 2 y 3 de madurez según el modelo ITS-CMM (Niessink y Van Vliet, 1999b).
- Poder registrar múltiples evaluaciones del mismo proceso o de procesos diferentes.
- Capacidad de definir nuevos cuestionarios.
- Poder editar y modificar los cuestionarios.
- Dado el carácter confidencial de la información manejada, se definen dos tipos de usuarios: administradores y evaluadores. Sólo los primeros estarán capacitados para editar cuestionarios, añadir nuevos cuestionarios o consultar las evaluaciones realizadas por otros usuarios.

Posteriormente se comprobó que MAN-Quest podría ser mucho más útil si se integraba plenamente en el Entorno MANTIS. Esto permitiría aprovechar las capacidades de genericidad que aporta MANTIS de cara a incluir nuevas funcionalidades como permitir definir cualquier tipo de cuestionario (que se representan como modelos de cuestionarios del nivel M1); o poder asociar a un mismo cuestionario diversos modelos de evaluación (que también se representan como modelos del nivel M1).

Para poder hacer esta abstracción mayor en la herramienta ha sido necesario el rediseño del anterior prototipo, incluyendo la elaboración de un “metamodelo de cuestionario”, perteneciente al nivel M2 de la arquitectura conceptual de MANTIS, que contiene clases como `datos_de_cabecera`, `pregunta`, `respuesta`, `sesión`, `resultado_global`, etc. MAN-Quest ha sido desarrollado como proyecto fin de carrera por la alumna María del Mar Jiménez (Jiménez, 2001).

7.3.6.1. Casos de Uso.

Siguiendo las especificaciones anteriores se deduce que existen dos actores: administrador y encuestado. El administrador es el responsable de la gestión de los cuestionarios y de analizar los resultados obtenidos con los mismos. Los encuestados simplemente responden a las preguntas de los cuestionarios.

Como se puede observar en la Figura 7-22, existen cinco casos de uso principales: “Crear Cuestionario”, “Modificar Cuestionario”, “Borrar Cuestionario”, “Analizar Resultados” y “Evaluar”. También existe un caso de uso “Validar Contraseña” que es usado por los anteriores para satisfacer el requisito de seguridad y confidencialidad preguntando el nombre de usuario y su clave. El caso de uso “Crear Cuestionario” se extiende a los casos “Crear Pregunta” y “Crear Respuesta”. De igual forma, el caso de uso “Modificar Cuestionario” engloba a los casos “Modificar Pregunta” y “Modificar Respuesta”. El caso de uso “Evaluar”, el único realizable por los actores de tipo encuestado, se extiende con los casos de uso “Crear Empresa” y “Crear Empleado”, que sirven para añadir una nueva organización a evaluar y para añadir un nuevo evaluador.

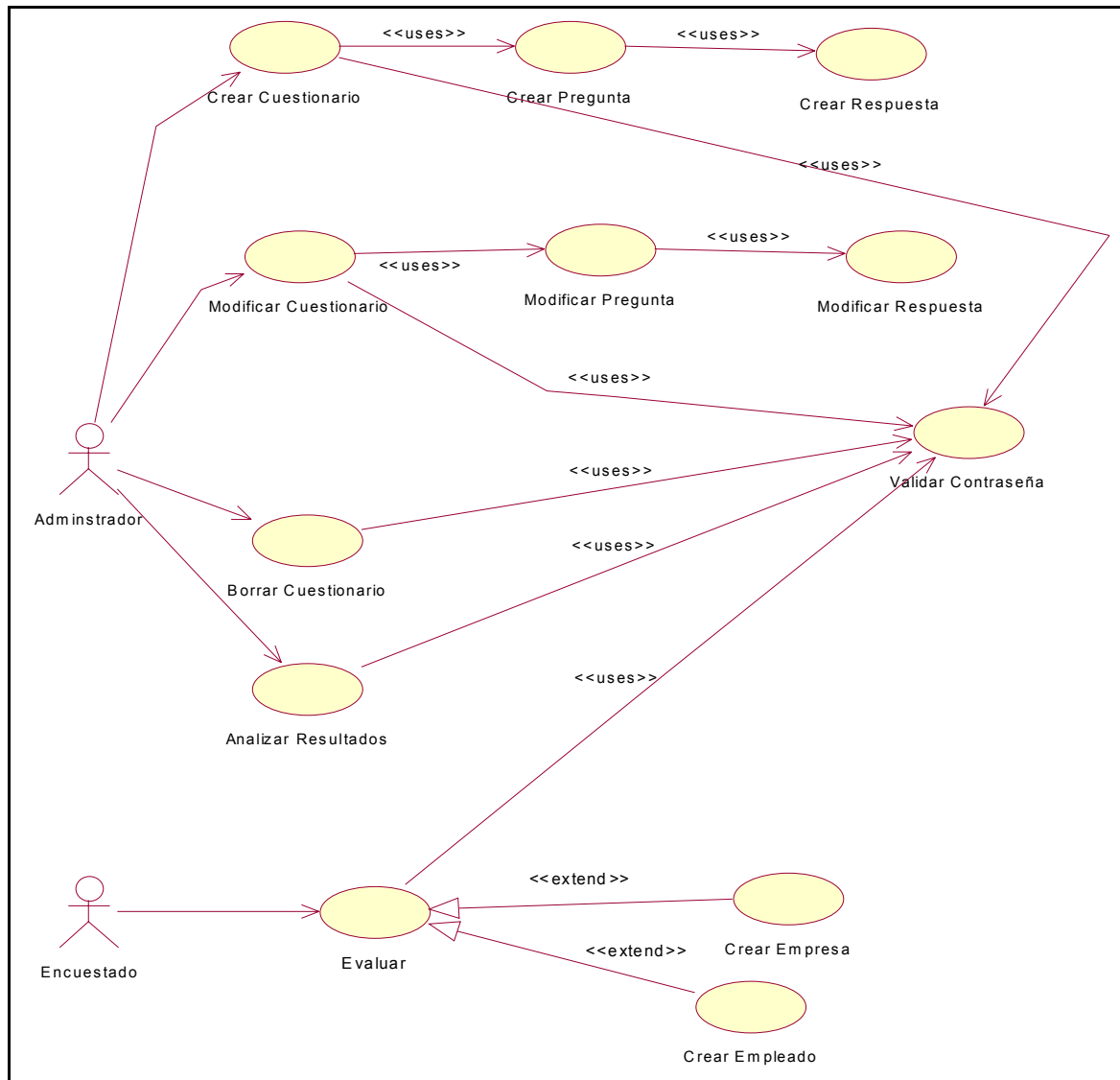


Figura 7-22. MAN-Quest: Diagrama UML de casos de uso.

7.3.6.2. Interfaz de Usuario.

Las características de la aplicación hacen que el interfaz de usuario y el modo de uso sean bastante sencillos. La aplicación comienza con un formulario de control que pregunta el usuario y la clave de acceso. En función de la respuesta se activarán los casos de uso del administrador o del encuestador que aparecen como opciones en la barra de herramientas superior.

Con la opción “Crear cuestionario” se abre un formulario que pregunta diversos datos del cuestionario (nivel de madurez que se evalúa, número de respuestas por preguntas, etc.) y que también sirve para ir introduciendo las sucesivas preguntas y respuestas (ver Figura 7-23). El formulario para “Modificar cuestionario” es similar al anterior, mientras que el formulario para “Borrar cuestionario” simplemente pregunta los identificadores del cuestionario o de la pregunta que se desea eliminar.

Figura 7-23. MAN-Quest: Formulario para crear cuestionarios.

En la Figura 7-24 se muestra el formulario para responder a un cuestionario. Por último, para calcular y mostrar la evaluación del nivel de madurez, correspondiente a un cuestionario ya completado, se debe elegir la opción de menú “Análisis de Resultados”. El usuario (de tipo administrador) puede asignar distintas ponderaciones a los distintos tipos de respuestas. Pulsando el botón evaluar se muestra un mensaje con el resultado de la evaluación. También es posible visualizar el resultado en forma de gráficos de diversos tipos.

Figura 7-24. MAN-Quest: Formulario para rellenar cuestionario.

“Experiencia es el nombre que todos dan a sus errores” (Oscar Wilde).

8. Conclusiones.

En este capítulo final se realiza un balance global de los resultados del trabajo llevado a cabo en esta tesis. Con este fin, los apartados incluidos han sido los siguientes: análisis de la consecución de objetivos y de los principales resultados que los corroboran, contraste de los mismos en publicaciones científicas, y líneas de trabajo abiertas.

8.1. Análisis de la Consecución de Objetivos.

Para analizar el grado de cumplimiento de los objetivos parciales (OP) y del objetivo general (OG) formulados en el capítulo 1, se ha utilizado el método de listas de comprobación de los productos de trabajo (PT) asociados con cada objetivo. Este método tiene la ventaja de que, a la vez se están indicando las principales aportaciones de la tesis. De hecho, los productos de trabajo que se incluyen a continuación son aportaciones que pueden ser comprobadas por la evidencia, bien mediante un documento o sub-documento escrito, bien mediante un artefacto software (prototipo de una herramienta). Las principales aportaciones de esta tesis, a juicio del autor, se han resaltado en negrita.

8.1.1. Objetivos Parciales.

A continuación se enumeran, para cada OP, los productos y sub-productos de trabajo que le corresponden. Al lado del código asignado a cada PT, se indica entre paréntesis el capítulo de la tesis que documenta dicho producto). El símbolo “↔” significa un PT que se plantea dentro de las líneas futuras que la tesis deja abiertas.

OP.1: Estudiar las tecnologías software y, en particular, las llamadas “tecnologías de proceso software”, útiles para gestionar de forma rigurosa el mantenimiento de software.

- ➡ PT 1.a (capítulo 3): Estudio global de las tecnologías de proceso software y, en particular, de las propuestas de arquitecturas para EIS y de catálogo de servicios EIS.
- ➡ PT 1.b (capítulo 3): Estudio de los lenguajes y entornos para el modelado y reificación de procesos.
- ➡ PT 1.c (capítulo 3): Estudio de los estándares y propuestas de ciclo de vida y de modelos de procesos útiles para el mantenimiento de software y su gestión.

OP.2: Estudiar las propuestas más actualizadas de métodos y técnicas para la gestión del mantenimiento de software.

- ➡ PT 2.a (capítulo 4): Estudio bibliográfico de las principales publicaciones sobre el tema, incluyendo el análisis de más de 700 referencias.
- ➡ PT 2.b (capítulo 4): Estado del arte con las propuestas más significativas de los últimos años para la gestión del mantenimiento de software.

OP.3: Ampliar la metodología MANTEMA con una colección de métodos, técnicas y guías útiles en la gestión de proyectos de mantenimiento de software.

- ➡ PT 3.a (capítulo 6): colección de procedimientos para los procesos de gestión. Consta de los siguientes sub-productos:
 - ➡ PT 3.a.1: **Versión 3 de la metodología MANTEMA** incluyendo una nueva definición de actividades iniciales y finales comunes, y nuevos documentos, especiales **para la externalización de servicios de mantenimiento**.
 - ➡ PT 3.a.2: **Propuesta de mejora basada en la gestión del conocimiento** mediante la incorporación de un sistema basado en agentes en el Entorno MANTIS.

- ➡ PT 3.a.3: Modelo económico para la **planificación de los recursos humanos necesarios para el mantenimiento correctivo urgente**.
- ↔ PT 3.a.4: Ecuaciones empíricas para la predicción del esfuerzo de mantenimiento de aplicaciones antiguas (legacy code).
- ➡ PT 3.a.5: Definición de los **objetivos de control en proyectos de mantenimiento**, basada en una Adaptación de la propuesta CobiT para la auditoría de sistemas de información.
- ➡ PT 3.a.6: Técnica para la **identificación y priorización de los riesgos en la externalización de servicios de mantenimiento**.
- ➡ PT 3.b (capítulo 6): colección de procedimientos para los procesos de la organización. Consta de los siguientes sub-productos:
 - ➡ PT 3.b.1: Cuestionario **guía para evaluar la madurez** (niveles 2 y 3) **de un servicio de mantenimiento**, basado en el modelo ITS-CMM de madurez de servicios de tecnologías de la información.
 - ↔ PT 3.b.2: **Propuesta para la inclusión en el Entorno MANTIS de un modelo para la evaluación del proceso de mantenimiento** basado en ISO 15504, incluyendo la definición de la estructura del artefacto “resultado de la evaluación”.
 - ➡ PT 3.b.3: Técnica para la **distribución de los recursos humanos** entre las peticiones de modificación **de una cartera de proyectos de mantenimiento**, basada en un modelo de análisis costes-beneficios restringido.
 - ↔ PT 3.b.4: **Propuesta para la inclusión en el Entorno MANTIS de un modelo genérico para la medición** basado en la ontología y el metamodelo de la medida (capítulo 5), que integre las métricas de la metodología MANTEMA y los indicadores de nivel de servicio.

OP.4: Desarrollar prototipos de herramientas software para los procedimientos del OP.3.

- ➡ PT 4.a (capítulo 7): colección de prototipos de herramientas horizontales del Entorno MANTIS. Consta de los siguientes sub-productos:
 - ↔ PT 4.a.1: Prototipo del interfaz de integración MANTIS-Tool.
 - ➡ PT 4.a.2: Prototipo del **componente RepManager para la gestión del repositorio integrado** del Entorno **basado en XML/XMI**, incluido el componente “Implementación MOF” para el manejo de modelos y metamodelos MOF.
 - ↔ PT 4.a.3: **Propuesta de arquitectura del gestor de la base de conocimiento KM-MANTIS, basado en tres comunidades de agentes**.
- ➡ PT 4.b (capítulo 7): colección de prototipos de herramientas verticales del Entorno MANTIS. Consta de los siguientes sub-productos:
 - ➡ PT 4.b.1: **Prototipo MANTOOL para la gestión de peticiones de modificación** con la metodología MANTEMA u otros modelos alternativos.
 - ➡ PT 4.b.2: Prototipo METAMOD de **herramienta para la administración de modelos y metamodelos basados en MOF**, incluyendo la gestión de las correspondencias entre elementos de diferentes niveles de abstracción y de la importación y exportación de/hacia otras herramientas CASE de diseño.
 - ➡ PT 4.b.3: **Prototipo MANTICA para el cálculo y visualización de métricas de mantenibilidad** de esquemas de bases de datos relacionales, objeto-relacionales y conceptuales.

- ➡ PT 4.b.4: **Prototipo SREM para la planificación (estimación) de los recursos humanos necesarios para el mantenimiento correctivo urgente** (basado en el PT 3.a.3).
- ➡ PT 4.b.5: **Prototipo CREM para la distribución de los recursos humanos** entre las peticiones de modificación **de una cartera de proyectos de mantenimiento** (basado en el PT 3.b.3).
- ➡ PT 4.b.6: **Prototipo MAN-Quest para gestionar y rellenar cuestionarios guía para evaluar la madurez de servicios de mantenimiento** (basado en el PT 3.b.1).

OP.5: Definir una arquitectura software adecuada para dar soporte automático integrado a los prototipos del OP.4.

- ➡ PT 5.a (capítulo 7): **Arquitectura software del Entorno MANTIS** estableciendo tres categorías de herramientas (internas horizontales, internas verticales, y externas) y los requisitos tecnológicos para la integración en las dimensiones de datos, control, presentación, procesos y marco de trabajo.
- ↔ PT 5.b (capítulo 7): **Modelo de uso de Sistemas de Gestión de Flujos de Trabajo** externos durante las fases de diseño (definición de los procesos) y ejecución (reificación) de los procesos. Incluye los modelos de interacción MANTIS-SGFT basados en repositorio compartido y en paso de mensajes con el nuevo estándar XPDL.

OP.6: Definir un marco de trabajo conceptual que ayude a gestionar proyectos de mantenimiento de software de forma integrada.

- ➡ PT 6.a (capítulo 5): **Arquitectura conceptual** de cuatro niveles (datos, modelos, metamodelos y meta-metamodelos) **basada en el estándar MOF**.
- ➡ PT 6.b (capítulo 5): **Sistema de procesos** para el Entorno MANTIS basado en los estándares ISO 12207 y 15504.
- ➡ PT 6.c (capítulo 5): Colección de ontologías para la compartición del conocimiento del dominio del problema (la gestión de proyectos de mantenimiento) entre los agentes humanos y las herramientas software, y propuesta razonada de cómo utilizarlas en este o en otros EIS. Consta de los siguientes sub-productos:
 - ➡ PT 6.c.1: **Ontología para el mantenimiento**, formada por las sub-ontologías de los productos, de las actividades, de organización del proceso (formada por procedimientos, gestión de peticiones, y problemas), y de los agentes.
 - ➡ PT 6.c.2: **Ontología de los flujos de trabajo**, incluyendo un modelo conceptual de los flujos de trabajo basado en el modelo de referencia de la “*Workflow Management Coalition*”, y modelos de estados para proyectos y actividades.
 - ➡ PT 6.c.3: **Ontología de la medida**, incluyendo un modelo conceptual del proceso de medición.
- ➡ PT 6.d (capítulo 5): Colección de metamodelos para ayudar a implementar, utilizando la arquitectura conceptual multinivel, las ontologías y el sistema de procesos en el repositorio de MANTIS. Consta de los siguientes sub-productos:
 - ➡ PT 6.d.1: **Metamodelo genérico de proceso software**, que está formado por tres paquetes: básico, de los flujos de trabajo, y organizacional.
 - ➡ PT 6.d.2: **Metamodelo de la medida** (basado en el PT 6.c.3).

El autor opina que se han alcanzado satisfactoriamente los seis objetivos parciales de la tesis.

8.1.2. Objetivo Global.

Recordemos el objetivo global de este trabajo:

Definir un entorno extendido de ingeniería del software que permita abordar de forma integrada (desde una perspectiva amplia de proceso de negocio), la gestión de proyectos de mantenimiento del software.

Del anterior análisis de los OP ya cabe deducir que el objetivo global ha sido razonablemente logrado. Esta afirmación se refuerza con el análisis adicional de los PT de carácter global, es decir, de los resultados que no se corresponden con alguno de los OP anteriores sino con el objetivo global. Estos son los PT globales obtenidos:

- ➡ PT G.1 (capítulos 1 y 5): **Definición de un nuevo concepto:** Un **EIS extendido** es “*una colección de herramientas conceptuales, metodológicas y software para abordar proyectos software desde una perspectiva amplia de proceso de negocio, integrando los procesos de ingeniería del software y los procesos de gestión y organizacionales*”.
- ➡ PT G.2 (capítulo 5): **Identificación de las principales características deseables en un EIS extendido:** integración por medio de herramientas, orientación a procesos, especialización, y escalabilidad y adaptabilidad.
- ➡ PT G.3 (capítulo 5): **Identificación de los componentes básicos para definir el marco de trabajo de un EIS extendido:** arquitectura conceptual multinivel, sistema de procesos, ontologías y, opcionalmente, metamodelos.
- ➡ PT G.4 (capítulos 1 y 5): **Definición de la estructura general** (catálogo de herramientas conceptuales, metodológicas y software) **del Entorno MANTIS** como EIS extendido especializado en la gestión de proyectos de mantenimiento de software.

Estos cuatro PT han sido completados y, por tanto, el objetivo global de la tesis ha sido alcanzado.

Una matización al PT G.4 es que, por su propia naturaleza, este PT se refiere a la lista actual de elementos del Entorno MANTIS, ya que siempre será posible idear nuevos procedimientos y herramientas para ayudar a gestionar proyectos de mantenimiento. En esto, el límite sólo está en la propia imaginación.

Para acabar este análisis de la consecución de objetivos y de los principales resultados de la tesis, este autor desea resaltar que, en su opinión, las ideas planteadas en los PT generales y en los componentes del marco de trabajo conceptual del Entorno MANTIS son útiles para abordar los proyectos de mantenimiento desde un punto de vista diferente al tradicional de los ingenieros de software, más global y más integrador, incluso aunque no se utilice absolutamente nada más (ni métodos ni aplicaciones software) de los resultados y propuestas de este trabajo. Tal como se explicó en el capítulo 1, el Entorno MANTIS pretende ser una aportación para que

la capacidad de las organizaciones que realizan mantenimiento de software pueda alcanzar los niveles 3 y 4 del modelo de capacidad ISO 15504 (ISO/IEC, 1998c).

Además, las ideas generales plasmadas en los PT globales son trasladables a la gestión de otros proyectos software, diferentes de los orientados a servicios de mantenimiento.

8.2. Contraste de Resultados.

Diferentes resultados y propuestas de esta tesis han sido publicados en diversos foros científicos. En la Tabla 8-1 se muestra una pequeña estadística de las publicaciones realizadas, que están enmarcadas dentro del ámbito de la tesis. Las publicaciones sometidas a evaluación en estos momentos se han indicado con un signo “+”.

Tipo	Internacional	Nacional o Latinoamericana	Total
Artículos en revistas SCI	2+1		2+1
Artículos en otras revistas	1	0+1	1+1
Capítulos de libros	5		5
Congresos de nivel alto (A)	6		6
Congresos de nivel medio (B)	2	6	8
Congresos de nivel bajo (C)	2	7	8
TOTAL	18+1	13+1	31+2

Tabla 8-1. Breve estadística de las publicaciones de la tesis.

Los artículos en revistas han sido divididos en dos grupos según tengan o no índice de impacto en el “*SCI - Science Citation Index*” (ISI, 2003). Los congresos, conferencias, jornadas y *workshops* también se han clasificados en tres categorías: A para aquellos internacionales que aparecen en el SCI o en el “*Citation Index for Computer Science*” del CiteSeer (NEC, 2002), B para los que tuvieron una ratio de aceptación inferior al 40%, y C para el resto.

En la Tabla 8-2 se resumen las publicaciones clasificadas en función de la temática, dentro del trabajo de la tesis, que abordan de manera central. Para identificar cada publicación se ha utilizado un código (año y siglas) que se detalla en las dos tablas posteriores.

Tema / Sub-tema	Publicaciones
El Entorno MANTIS	
Características y estructura general	2002-Addison, 2002-Idea, 2001-JISBD
Marco de Trabajo Conceptual	
Ontologías	2003-IJSEKE, 2003-SEKE
Arquitectura conceptual	2002-ICEIS, 2001-ISSI, 2001-ISE
Metamodelos basados en MOF	2001-ICIE
Ampliar la Metodología MANTEMA	
Integrar la externalización de servicios de mantenimiento	2002-ITM
Propuesta de incorporar nuevos métodos de gestión	2000-CINTAI

Tema / Sub-tema	Publicaciones
Procedimientos para Gestionar el Mantenimiento	
Objetivos de control y auditoría para el mantenimiento	2000-Idea, 1999-CIIC, 1999-CASI, 2000-SIPCVISTI
Evaluación y mejora del proceso de mantenimiento	2002-Kluwer, 2003-CyS, 2003-IDEAS
Evaluación de la madurez de un servicio de mantenimiento	2001-PROFES
Predecir el esfuerzo de mantenimiento de “ <i>legacy code</i> ”	2001-ICSM
Planificación del mantenimiento correctivo no urgente	2000-ESCOM
Mejora basada en la gestión del conocimiento	2002-IWPAAMS
Identificación y priorización de riesgos	2000-JISBD
Integración de flujos de trabajo	2001-IDEAS, 2001-CACIC
Herramientas Software	
MANTOOL	2001-JSME
METAMOD	2001-WSES, 2001-Bitworld, 2001-ECC
RepManager	2002-PROFES, 2001-CLEI
Método de Trabajo	
Uso de Investigación-Acción	2002-SPE, 2002-MIFISIS

Tabla 8-2. Lista de publicaciones clasificadas por temas.

8.2.1. Publicaciones Internacionales.

Código	Referencia	Publicación
Artículos en revistas SCI:		
2001-JSME	Polo et al, 2001d	Polo, M., Piattini, M. y Ruiz, F. (2001d): MANTOOL: a Tool for Supporting the Software Maintenance Process. <i>Journal of Software Maintenance and Evolution: Research and Practice</i> . 13(2), pp. 77-95, John Wiley & Sons. Factor de impacto CSI=0'971.
2002-SPE	Polo et al, 2002a	Polo, M., Piattini, M. y Ruiz, F. (2002a): Using a Qualitative Research Method for Building a Software Maintenance Methodology. <i>Software Practice & Experience</i> ., 32(13), pp. 1239-1260. John Wiley & Sons. Factor de impacto CSI=0'571.
2003-IJSEKE	Ruiz et al, 2003	Ruiz, F. , Vizcaíno, A., Piattini, M. y García, F. (2003): An Ontology for the Management of Software Maintenance Projects. <i>International Journal of Software Engineering and Knowledge Engineering</i> (pendiente de evaluación). Factor de impacto CSI=0'190.
Artículos en otras revistas:		
2002-ITM	Polo et al, 2002b	Polo, M., Piattini, M. y Ruiz, F. (2002b): Integrating Outsourcing in the Maintenance Process. <i>Information Technology and Management</i> , Vol 3(3), pp. 247-269. Kluwer Academic Publishers.
Capítulos de libros:		
2002-Addison	Ruiz et al, 2002a	Ruiz, F. , Piattini, M. y Polo, M. (2002a): An Integrated Environment for Managing Software Maintenance Projects. En “ <i>The Guide to IT Service Management, volume I</i> ”, chapter 31, pp. 567-588. Addison-Wesley , ISBN 0201737922.
2000-Idea	Ruiz et al, 2000a	Ruiz, F. , Piattini, M., Polo, C. y Calero, C. (2000a): Audit of Software Maintenance Process. En “ <i>Auditing Information Systems</i> ”. Idea Group Publishing (Estados Unidos), pp. 67-108.

2002-Idea	Ruiz et al, 2002b	Ruiz, F. , García, F., Piattini, M. y Polo, M. (2002b): Environment for Managing Software Maintenance Projects. En “ <i>Advances in Software Maintenance Management: Technologies and Solutions</i> ”. Idea Group Publishing (Estados Unidos), capítulo X, pp. 255-290. ISBN 1-59140-047-3.
2002-Kluwer	García et al, 2002a	García, F., Ruiz, F. , Piattini, M. y Polo, M. (2002a): Conceptual Architecture for the Assessment and Improvement of Software Maintenance. En “ <i>Enterprise Information Systems IV</i> ”. Kluwer Academic Publishers (Países Bajos), pp. 219-226. ISBN 1-4020-1086-9.
2001-WSES	García et al, 2001a	García, F., Márquez, L., Ruiz, F. , Piattini, M. y Polo, M. (2001a): A Tool for the Management of the Software Maintenance Process. En “ <i>Advances in Signal Processing and Computer Technologies</i> ”. WSES Press (Estados Unidos), ISBN 960-8052-37-8, pp. 228-232.
Congresos de nivel A:		
2002-PROFES	Ruiz et al, 2002c	Ruiz, F. , Piattini, M., García, F. y Polo, M. (2002c): An XMI-based Repository for Software Process Meta-modeling. <i>Product Focused Software Process Improvement (PROFES’2002)</i> . Rovaniemi (Finlandia), 9-11 diciembre. Springer-Verlag, LNCS 2559 , ISBN 3-540-00234-0. pp. 546-558.
2001-PROFES	Polo et al, 2001c	Polo, M., Piattini, M., Ruiz, F. y Jiménez, M. (2001c): Assessment of Maintenance Maturity in IT Departments of Public Entities: Two Case Studies. <i>Product Focused Software Process Improvement (PROFES’2001)</i> . Springer-Verlag, LNCS 2188 , ISBN 3-540-42571-3, pp. 86-97.
2003-SEKE	Género et al, 2003	Género, M., Ruiz, F. , Piattini, M., Calero, C. y García, F. (2003): An Ontology for Software Measurement. <i>International Conference on Software Engineering and Knowledge Engineering (SEKE’2003)</i> . San Francisco, CA (USA), julio-2003 (aceptado).
2001-ICSM	Polo et al, 2001b	Polo, M., Piattini, M. y Ruiz, F. (2001b): Using Code Metrics to Predict Maintenance of Legacy Programs: a Case Study. <i>IEEE International Conference on Software Maintenance (ICSM’2001)</i> . Florence (Italia), pp. 202-208.
2002-ICEIS	García et al, 2002b	García, F., Ruiz, F. , Piattini, M. y Polo, M. (2002b): Conceptual Architecture for the Assessment and Improvement of Software Maintenance. <i>4th International Conference on Enterprise Information Systems (ICEIS’02)</i> . 2-6 abril, Ciudad Real (España), pp. 610-617.
2000-ESCOM	Polo et al, 2000a	Polo, M., Piattini, M. y Ruiz, F. (2000a): Planning the non-Planneable Maintenance. <i>11th European Software Control and Metrics Conference / 3rd Scope Conference on Software Product Quality (ESCOM-SCOPE 2000)</i> . Munchen (Germany). Shaker Publishing, pp. 49-57.
Congresos de nivel B:		
2001-ISSI	Ruiz et al, 2001c	Ruiz, F. , Piattini, M., y Polo, M. (2001c): An Conceptual Architecture Proposal for Software Maintenance. <i>13th International Symposium on System Integration (ISSI’01)</i> . Baden-Baden (Germany), pp. VIII:1-8.
2002-IWPAAMS	Vizcaíno et al, 2002a	Vizcaíno, A., Ruiz, F. , Favela, J. y Piattini, M. (2002a): A Multi-Agent Architecture for Knowledge Management in Software Maintenance. <i>I International Workshop on Practical Applications of Agents and Multiagent Systems (IWPAAMS’2002)</i> . Salamanca (España), 23-25 octubre.

Código	Referencia	Publicación
Congresos de nivel C:		
2001-ISE	Márquez et al, 2001a	Márquez, L., García, F., Ruiz, F. , Piattini, M. y Polo, M. (2001a): Managing Complexity of Software Processes. <i>International Symposium on Information Systems and Engineering</i> (ISE'2001). Las Vegas (USA), pp. 180-186.
2001-BitWorld	Ruiz et al, 2001d	Ruiz, F. , García, F., Márquez, L., Piattini, M. y Polo, M. (2001d): Tool based on MOF for Software Process Metamodelling. <i>Business Information Technology Management</i> (BitWorld'2001). El Cairo (Egipto), junio. CD-ROM.

Tabla 8-3. Lista de publicaciones internacionales enmarcadas dentro del trabajo de la tesis.

8.2.2. Publicaciones Nacionales y Latinoamericanas.

Código	Referencia	Publicación
Artículos en revistas:		
2003-CyS	García et al, 2003a	García, F., Ruiz, F. y Piattini, M. (2003a): Metamodelado y Medición para la Evaluación y Mejora del Proceso de Mantenimiento del Software. <i>Computación y Sistemas</i> , IPN-México (pendiente de evaluación).
Congresos de nivel B:		
2001-JISBD	Ruiz y Piattini, 2001	Ruiz, F. y Piattini, M. (2001): Entorno Global para la Gestión del Proceso de Mantenimiento del Software. <i>VI Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2001)</i> . Almagro (Ciudad Real), 21-23 noviembre, pp. 157-170.
2000-JISBD	Polo et al, 2000b	Polo, M., Piattini, M. y Ruiz, F. (2000b): Cuestionario para la Identificación de Riesgos en Proyectos de Mantenimiento. <i>V Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2000)</i> . Valladolid (España), pp. 47-52.
2001-CLEI	García et al, 2001b	García, F., Ruiz, F. , Piattini, M., Márquez, L. y Polo, M. (2001b): Propuesta de Repositorio basado en XMI para Metamodelado de Procesos Software. <i>XXVII Conferencia Latinoamericana de Informática (CLEI'01)</i> . Mérida (Venezuela), 24-28 septiembre, pp.70.
2001-IDEAS	Ruiz et al, 2001a	Ruiz, F. , Piattini, M., García, F. y Polo, M. (2001a): Metamodelos y Flujos de Trabajo para la Gestión del Proceso de Mantenimiento del Software. <i>4ª Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes de Software</i> (IDEAS'2001). Heredia (Costa Rica), abril-2001, pp. 132-142.
2001-ECC	Márquez et al, 2001c	Márquez, L., García, F., Ruiz, F. , Piattini, M. y Polo, M. (2001c): Herramienta para Metamodelización de Procesos Software. <i>Jornadas Chilenas de Computación</i> (ECC'01). Punta Arenas (Chile), noviembre. CD-ROM.
1999-CIIC	Ruiz et al, 1999b	Ruiz, F. , Piattini, M., Polo, M. y Calero, C. (1999b): Propuesta de un Marco Formal para la Auditoría del Proceso de Mantenimiento del Software. <i>6º Congreso Internacional de Investigación en Ciencias Computacionales</i> (CIIC'99). Cancún (México), septiembre. pp. 192-202.

Código	Referencia	Publicación
Congresos de nivel C:		
2003-IDEAS	García et al, 2003b	García, F., Ruiz, F. , Cruz, J.A. y Piattini, M. (2003b): Integración del Metamodelado y la Medición para la Mejora de los Procesos Software. <i>6° Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software</i> (IDEAS'2003). Asunción (Paraguay), 30-abril/2-mayo.
2002-MIFISIS	Ruiz et al, 2002d	Ruiz, F. , Polo, M. y Piattini, M. (2002d): Utilización de Investigación-Acción en la Definición de un Entorno para la Gestión del Proceso de Mantenimiento del Software. <i>I Workshop en Métodos de Investigación y Fundamentos Filosóficos en Ingeniería del Software y Sistemas de Información</i> (MIFISIS'2002). El Escorial (España), 18-nov., pp. 48-64.
2001-CACIC	Ruiz et al, 2001b	Ruiz, F. , Piattini, M., y Polo, M. (2001b): Using Metamodels and Workflows in a Software Maintenance Environment. <i>VII Argentine Congress on Computer Science</i> (CACIC'01). El Calafate (Argentina), octubre-2001.
1999-CASI	Ruiz et al, 1999a	Ruiz, F. , Piattini, M., Polo, M. y Calero, C. (1999a): Auditoría del Mantenimiento del Software: Propuesta de Objetivos de Control. <i>II Congreso Nacional de Auditoría y Control de Sistemas de Información</i> (CASI'99). Valencia (España), octubre. pp. 128-143.
2001-ICIE	Márquez et al, 2001b	Márquez, L., García, F., Ruiz, F. , Piattini, M. y Polo, M. (2001b): Herramienta para la Representación de Procesos Software mediante Modelos MOF. <i>VII Congreso Internacional de Ingeniería Informática</i> (ICIE'2001). Buenos Aires (Argentina), abril.
2000-CINTAI	Polo et al, 2000c	Polo, M., Garbajosa, J., Piattini, M. y Ruiz, F. (2000c): Métodos y Técnicas para la Mejora del Proceso de Mantenimiento. <i>VII Congreso Internacional de Nuevas Tecnologías y Aplicaciones Informáticas</i> . La Habana (Cuba). MIC, CD-ROM.
2000-SIPCVISTI	Ruiz et al, 2000b	Ruiz, F. , Piattini, M. y Polo, M. (2000b): Objetivos de Control para la Auditoría del Proceso de Mantenimiento del Software. <i>V Seminario Iberoamericano de Protección contra Virus Informáticos y Seguridad de las TI</i> . La Habana (Cuba), CD-ROM.

Tabla 8-4. Lista de publicaciones nacionales y latinoamericanas enmarcadas dentro del trabajo de la tesis.

8.2.3. Otras Publicaciones sobre Mantenimiento.

Tal como se comentó en capítulo 1, el autor de esta tesis ha estado trabajando en el tema de mantenimiento desde varios años antes de empezar a abordar los objetivos de la tesis. Esto supuso la obtención de diversas publicaciones, la mayoría dedicadas a divulgar el problema del mantenimiento o al desarrollo de la metodología MANTEMA. En la Tabla 8-5 se relacionan estas otras publicaciones.

Tipo	Publicación
Internacionales:	
Artículos en revistas	Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999b): Roles in the Maintenance Process. <i>ACM Software Engineering Notes</i> ; 24(4), pp. 84-86.

Libros editados	Polo, M., Piattini, M. y Ruiz, F. [editores] (2002): <i>Advances in Software Maintenance Management: Technologies and Solutions</i> . Idea Group Publishing (Estados Unidos), ISBN 1-59140-047-3.
Capítulos de libros	Polo, M., Piattini, M. y Ruiz, F. (2000): Managing the Software Maintenance Process. In “ <i>World Class IT Service Management Guide</i> ”. Ten Hagen & Stam Publishers (Países Bajos), ISBN 90-76383-46-4, pp. 213-223.
	Polo, M., Piattini, M. y Ruiz, F. (2000): Techniques and Methods for Managing the Maintenance Process. In “ <i>Signal Processing, Communications and Computer Science</i> ”. World Scientific and Engineering Society Press (Grecia), ISBN 960-8052-18-1, pp. 305-310.
	Polo, M., Piattini, M. y Ruiz, F. (2001): Experience adapting ISO/IEC 12207 to the maintenance process. In “ <i>Software Engineering – Software Standardization</i> ”. Dintel (España), ISBN 84-931933-2-1, pp. 165-189.
	Polo, M., Piattini, M. y Ruiz, F. (2002): A Methodology for Software Maintenance. En “ <i>Advances in Software Maintenance Management: Technologies and Solutions</i> ”. Idea Group Publishing (Estados Unidos), capítulo IX, pp. 228-254. ISBN 1-59140-047-3.
Congresos de nivel A	Piattini, M., Calero, C., Polo, M. y Ruiz, F. (1998): Maintainability in Object-Relational Databases. <i>First European Software Measurement Conference (FESMA’98)</i> . Antwerpen (Bélgica), mayo. pp. 223-230.
	Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999c): MANTEMA: A Complete Rigorous Methodology for Supporting Maintenance based on the ISO/IEC 12207 Standard. <i>Third Euromicro Conference on Software Maintenance and Reengineering (CSMR’99)</i> . Amsterdam (Países Bajos). IEEE Computer Society, pp. 178-181.
	Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999): MANTEMA: A Software Maintenance Methodology based on the ISO/IEC 12207 Standard. <i>4th IEEE International Software Engineering Standards Symposium (ISESS’99)</i> . Curitiba (Brazil). IEEE Computer Society , pp. 76-81.
	Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999): Using the ISO/IEC Tailoring Process for defining a Maintenance Process. <i>IEEE Conference on Standardization and Innovation on Information Technology (SIIT’99)</i> . Aachen (Germany). IEEE Computer Society , pp. 205-210.
	Polo, M., Piattini, M. y Ruiz, F. (2000): Improving the Quality of the Maintenance Process. <i>Second World Congress for Software Quality (2WCSQ)</i> . Yokohama (Japón), pp. 325-330.
	Ruiz, F. , Piattini, M., Polo, M. Y Calero, C. (1999): Maintenance Types in the MANTEMA Methodology. <i>International Conference on Enterprise Information Systems (ICEIS’99)</i> . Setubal (Portugal). pp. 192-202.
Congresos de nivel B	Piattini, M., Polo, M., Ruiz, F. y Calero, C. (1999): A Rigorous Approach for Software Maintenance. <i>Fourth World Conference on Integrated Design & Process Technology (IDPT’1999)</i> . Dallas (USA), junio-2000 (previsto inicialmente en Turquía en junio-1999). Society for Design and Process Science, CD-ROM.
Nacionales y Latinoamericanas:	
Artículos en revistas	Piattini, M., Ruiz, F. , Polo, M. y Calero, C. (1999): El Mantenimiento del Software. <i>Convergencia IT</i> ; nº 1, pp. 14-17. Fundación Dintel.
Libros escritos	Piattini, M., Ruiz, F. , Polo, M., Bastanchury, T., Fernández, I. y Martínez, M.A. (1998): “Mantenimiento del Software: Conceptos, Métodos, Herramientas y Outsourcing”. Ra-Ma , España, ISBN 84-7897-321-4, 266 pp.
	Piattini, M., Ruiz, F. , Polo, M., Villalba, J., Bastanchury, T., Martínez, M.A. y Nistal, C. (2000): <i>Mantenimiento del Software: Modelos, Técnicas y Métodos para la Gestión del Cambio</i> . Ra-Ma , España. ISBN 84-7897-448-2, 336 pp.
Capítulos de libros	Polo, M., Ruiz, F. y García, F. (2002): Calidad en Mantenimiento de Software. En “ <i>Calidad en el Desarrollo y Mantenimiento del Software</i> ”. Ra-Ma (España), ISBN 84-7897-544-6. Capítulo 13, pp. 271-304.

Tipo	Publicación
Congresos de nivel B	Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999): Adaptación de la norma ISO/IEC 12.207 al proceso de mantenimiento del software. <i>6º Congreso Internacional de Investigación en Ciencias Computacionales (CIIC'99)</i> . Cancún (México), septiembre. pp. 92-100.
	Ruiz, F. , Piattini, M., y Polo, M. (1998): MANTEMA: Una Metodología para la Gestión Integral del Mantenimiento del Software. <i>VI Encuentro Chileno de Computación (ECC'98)</i> . Antofagasta (Chile), noviembre. pp. 184-193.
Congresos de nivel C	Piattini, M., Polo, M., Ruiz, F. , Bastanchury, T., Fernández, I., Martínez, M.A. y Villalba, J. (1998): Calidad en el Mantenimiento de Software: Aplicación de las Normas ISO. <i>VII Congreso Español de la Calidad</i> . Madrid (España), junio. Asociación Española para la Calidad.
	Piattini, M., Calero, C., Polo, M. y Ruiz, F. (1999): Propuesta de Métricas para la Mantenibilidad de Bases de Datos ORACLE. <i>X Congreso Nacional de Usuarios de ORACLE (CUORE)</i> . A Coruña (España), octubre. CD-ROM.
	Piattini, M., Polo, M., Ruiz, F. y Calero, C. (1999): Utilización de los Estándares ISO/IEC en el Mantenimiento del Software. <i>V International Congress on Information Engineering (ICIE'99)</i> . Buenos Aires (Argentina). pp. 1-8.
	Piattini, M., Polo, M., Ruiz, F. y Calero, C. (1999): Proceso de Mantenimiento del Software. <i>I Jornadas Gallegas Universidad-Empresa de Ingeniería del Software</i> . A Coruña (España). pp 145-154.
	Ruiz, F. (1998): El Problema del Mantenimiento del Software: Importancia de la Mantenibilidad. <i>I Jornadas de Auditoría Informática (JAI'98)</i> . Ciudad Real (España), noviembre. pp. 245-268.
Informes técnicos	Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999a): <i>MANTEMA versión 2.0: una Metodología para el Mantenimiento de Software</i> . Informe Técnico UCLM-DI-99-01, Universidad de Castilla-La Mancha, Departamento de Informática, Grupo Alarcos, 91 pg.

Tabla 8-5. Lista de publicaciones internacionales enmarcadas dentro del trabajo de la tesis.

8.3. Líneas Abiertas de la Investigación.

A lo largo de los capítulos anteriores y, en especial, en el análisis de los resultados presentado en el apartado 8.1, se han manifestado varios aspectos de la propuesta de esta tesis factibles de mejora. También se han incluido propuestas de nuevos elementos del Entorno MANTIS. Todas ellas abren nuevas líneas de trabajo, actuales o futuras, que son comentadas a continuación.

8.3.1. Mejora Continua del Mantenimiento.

El proyecto TAMANSI, en vigor hasta diciembre de 2004 (ver capítulo 1), propone continuar los trabajos anteriores para incluir nuevas técnicas de gestión en los proyectos de mantenimiento de software y para lograr la mejora continua de los procesos de mantenimiento (nivel 5 optimizado del modelo de capacidad de ISO 15504). Esto último se aborda incidiendo en dos dimensiones de mejora: la gestión del conocimiento y la medición. A continuación se resumen estas dos nuevas líneas del trabajo de investigación.

8.3.1.1. Basada en la Gestión de Conocimiento.

En esta dimensión se trataría de explotar las posibilidades de los sistemas de gestión de conocimiento para la mejora de los procesos software en general, y del mantenimiento en particular. En este sentido existen algunas propuestas interesantes:

- Maurer y Dellen (1998) han propuesto una colección de requisitos para los sistemas de gestión de conocimiento orientados a procesos de desarrollo: soporte del trabajo síncrono, soporte del trabajo asíncrono, infraestructura de comunicación ubicua, acceso rápido y transparente al conocimiento sobre los procesos, gestión de configuración distribuida, repositorio para ontologías, flexibilidad en los procesos (soporte a diferentes modelos de proceso), y notificaciones de cambio pro-activas. Estas ideas son aplicables al mantenimiento de forma directa (y por tanto al Entorno MANTIS). Además, existe otro paralelismo entre esta propuesta y MANTIS: estos autores también asignan un papel importante a la tecnología de flujos de trabajo.
- Falbo et al (1999) sugieren utilizar servidores de conocimiento para mejorar la integración de las herramientas dentro de un EIS. Estos autores desarrollan su idea en el prototipo TABA, un “meta-EIS” que permite especificar e instanciar EIS específicos, incluyendo la definición de los procesos (sólo de desarrollo de software) y la selección de las herramientas.
- Las ideas de Fensel (2000) para integrar y gestionar conocimiento mediante el uso de ontologías pueden ser útiles para KM-MANTIS. Su propuesta de utilizar la norma “*Resource Description Framework*”(RDF) de la W3C (1999) debería ser analizada y comparada con XMI.

A partir de estas sugerencias y de los comentarios señalados en otras partes de la tesis, en esta línea de trabajo se han establecido ya las siguientes mejoras:

- a) Revisar la arquitectura de agentes de KM-MANTIS (PT 4.a.3), probablemente reduciendo las comunidades de agentes de productos y actividades a un solo agente. Estudiar cual de las dos alternativas es más eficiente.
- b) Implementar completamente KM-MANTIS incorporándole, tal como se detalló en el capítulo 7 (Vizcaíno et al, 2002), la generación de nuevo conocimiento en forma de reglas de gestión.

8.3.1.2. Basada en la Medición.

Esta segunda dimensión de mejora del mantenimiento está basada, principalmente, en las propuestas siguientes:

- El modelo de mejora del mantenimiento basado en la medición formulado por Niessink (2001), ya comentado en el capítulo 3.
- CMMi (*Capability Maturity Model Integration*), que destaca por su especial importancia. El objetivo fundamental que se persigue con CMMi es la integración de la mejora de los procesos y sistemas software. CMMi utiliza tres modelos tradicionales que son CMM-sw (el ya conocido CMM para el software), el modelo de Ingeniería de Sistemas EIA/IS 731, y el modelo de desarrollo integrado del producto y del proceso. CMMi incluye estos tres modelos en un modelo integrado de madurez de la capacidad. Dentro del contexto de CMMi, estos modelos proporcionan un conjunto común de

requerimientos de proceso que conducen a la utilización de las mejores prácticas y conocimientos dentro de una organización dedicada al desarrollo y/o mantenimiento del software. Para alcanzar los objetivos de cada nivel de madurez, la organización debe entender, controlar y mejorar de forma continua sus procesos. Como consecuencia de tener procesos efectivos y eficientes en una organización, se obtienen productos de alta calidad que satisfacen plenamente las necesidades del cliente y de la organización.

Las mejoras y líneas de trabajo planteadas en este apartado son las siguientes:

- a) Aplicar las dos propuestas anteriores para definir un marco para la evaluación de modelos de procesos, compatible e integrable dentro de la arquitectura conceptual, ontologías y metamodelos de MANTIS. Esta idea, que es la base de un nuevo proyecto de tesis doctoral, tiene una parte de implementación de un nuevo componente horizontal del Entorno MANTIS, llamado GenMETRIC (PT 3.b.4). Más detalles de esta línea de trabajo se pueden consultar en (García et al, 2003a).
- b) También se ha creado un grupo de trabajo formado por investigadores de varias universidades españolas (Málaga, Sevilla, Politécnica de Cataluña y Grupo Alarcos) para elaborar un modelo global para la medición del software (es decir, proceso de medición y métricas), revisando la ontología y metamodelo de la medida aquí propuestos (Género et al, 2003). Se pretende que esto sea el germen de una red científica española dedicada a la medición del software.

8.3.2. Mantenimiento Ágil del Software.

Esta última línea de trabajo abierta se ha originado como consecuencia de las demandas, probablemente debido a que es un tema que “está de moda”, de varias de las empresas con las que colabora el Grupo Alarcos, al que pertenece el autor de la tesis. Esto ha coincidido con la integración del Grupo en la subred española de la europea NAME (*Network of Agile Methodologies Experience*), una red que busca la excelencia y la colaboración de los investigadores europeos en métodos y técnicas para el desarrollo y mantenimiento ágil del software. El coordinador en España de esta subred es el profesor José Hilario Canós de la Universidad Politécnica de Valencia. El papel del grupo Alarcos será estudiar si estas ideas son trasladables, cómo y en qué medida, desde el desarrollo al mantenimiento.

Como consecuencia de lo anterior, se decidió presentar un proyecto nuevo a la última convocatoria del Ministerio de Ciencia y Tecnología. El principal objetivo de este proyecto, llamado MAS (Mantenimiento Ágil del Software) y presentado en el capítulo 1, es desarrollar entornos avanzados de generación automática de pruebas para mantenimiento ágil del software. Esta misma idea es la base de una nueva propuesta de tesis doctoral.

8.3.3. Mejorar la Arquitectura y Tecnología del Entorno.

Cuando en el capítulo 7 se ha presentado la arquitectura software del Entorno MANTIS y cuando, también, se han enumerado los actuales prototipos de herramientas software, se han sugerido algunas posibles mejoras:

- a) El repositorio de MANTIS utiliza archivos DTD (*Data Type Document*) asociados a los documentos XMI o XML. Una clara mejora técnica es sustituir dichos DTD por archivos basados en el nuevo estándar “XML Schema” de la W3C (2001), ya que los

esquemas XML tienen significativas ventajas, especialmente semánticas, frente a los DTD.

- b) Igual que la arquitectura del gestor de conocimiento KM-MANTIS utiliza un servidor XML para dar servicio de bajo nivel de almacenamiento, búsqueda y recuperación de documentos XML, se está estudiando la posibilidad de utilizar la misma alternativa en lugar de que el componente RepManager utilice DOM para el manejo de documentos XML. En la actualidad, a tal fin, se está probando el sistema “Tamino XML server 4.0” de Software AG. El problema es que dicha opción debe ser cuidadosamente estudiada para no ir en contra del principio de que los elementos internos de MANTIS (y el gestor del repositorio es un componente interno básico) no tengan una dependencia crítica de un elemento externo.
- c) Otra mejora de carácter técnico (ver PT 4.a.1) es implementar el interfaz de usuario interactivo de MANTIS-Tool. En la actualidad se está estudiando la posibilidad de utilizar “*web services*” como mecanismo adicional para la interacción entre las diversas herramientas de MANTIS y también con las herramientas externas. En esta línea, se está realizando un proyecto fin de carrera que intenta implementar una capa superior de servicios web para la comunicación entre el gestor del repositorio RepManager, la herramienta METAMOD de metamodelización y el interfaz de integración, MANTIS-Tool.
- d) Otra última mejora de carácter técnico es incorporar a MANTIS un componente para mejorar la interacción con Sistemas de Gestión de Flujos de Trabajo externos. Para ello, se está desarrollando una prueba consistente en la realización de un proyecto fin de carrera que implementa el servicio de paso de mensajes desde MANTIS-Tool, o directamente desde RepManager, hacia o desde un SGFT. El único requisito para la interacción es que el SGFT utilice el nuevo estándar XPDL (*XML Process Definition Language*) de la WfMC (2002).

Anexos.

- A. Publicaciones sobre Gestión del Mantenimiento.
- B. Metodología MANTEMA.
- C. MOF: Meta-Object Facility.
- D. XMI: XML Metadata Interchange.
- E. Formalismo REFSENO para Representación de Ontologías.
- F. SPEM: Software Process Engineering Metamodel.
- G. Correspondencias entre los Niveles Conceptuales de MANTIS.
- H. Manuales de Usuario.
- I. Ejemplos de Documentos DTD y XML.

A. Publicaciones sobre Gestión del Mantenimiento.

En la Tabla A-1 se muestra la lista completa de las publicaciones científicas relacionadas con la gestión del PMS entre los años 1998 y 2001, ambos incluidos. Estas referencias han sido obtenidas mediante un proceso de búsqueda y clasificación comentado al principio del capítulo 4, donde también se incluye un análisis estadístico de los resultados. Las referencias están clasificadas en función de los cuatro parámetros siguientes:

- *Medio*: tipo de medio utilizado en la búsqueda.
- *Fuente*: siglas de la revista o conferencia.
- *Dimensión*: aspecto de la gestión del PMS al que se refiere (según una clasificación basada en el modelo de procesos de ISO 15504-2).
- *Año*: año de la publicación (sólo se han considerado los años 1998-2001).

Publicaciones Relacionadas con la Gestión del Mantenimiento del Software (1998-2001)				
Medio ¹	Fuente ²	Dimensión ³	Año	Autores y Título
ESP	ICSM	CAL	1999	Norman F. Schneidewind; Software Quality Maintenance Model.
ESP	ICSM	GES	2001	Charles J. Poole, Tim Murphy, Jan Willem Huisman, Allen Higgins; Extreme Maintenance.
ESP	ICSM	GES	2001	Lerina Aversano, Andrea de Lucia, Silvio Stefanucci, Sergio Betti; Introducing Workflow Management in Software Maintenance Processes.
ESP	ICSM	HUM	2001	G. Antoniol, M. di Penta, G. Casazza, G.A. di Lucca, F. Rago; A Queue Theory-Based Approach to Staff Software Maintenance Centers.
ESP	ICSM	MED	1998	F. Niessink, H. Van Vliet; Two Case Studies in Measuring Software Maintenance Effort.
ESP	ICSM	MED	1998	M. Ramage, K. Bennett; Maintaining Maintainability.
ESP	ICSM	MED	1999	F. Fioravanti, P. Nesi, F. Stortoni; Metrics for Controlling Effort During Adaptive Maintenance of Object Oriented Systems.
ESP	ICSM	MED	2001	Macario Polo, Mario Piattini, Francisco Ruiz; Using Code Metrics to Predict Maintenance of Legacy Programs: A Case Study.
ESP	ICSM	MEJ	2001	Tracy Hall, Austen Rainer, Nathan Baddoo, Sarah Beecham; An Empirical Study of Maintenance Issues within Process Improvement Programmes in the Software Industry.
ESP	ICSM	MEJ	2001	Frank Niessink; Perspectives on Improving Software Maintenance.
ESP	ICSM	MOD	1998	H.-J. Kung, C. Hsu; Software Maintenance Life Cycle Model.
ESP	ICSM	MOD	2001	Keith Bennett, Malcolm Munro, Nicolas Gold, Paul Layzell, David Budgen, Pearl Brereton; An Architectural Model for Service-Based Software with Ultra Rapid Evolution.
ESP	ICSM	MOD	2001	Mira Kajko-Mattsson; Towards A Business Maintenance Model.
ESP	ICSM	PRO	2000	Taizan Chan; Beyond Productivity in Software Maintenance: Factors Affecting Lead Time in Servicing Users' Requests.

Publicaciones Relacionadas con la Gestión del Mantenimiento del Software (1998-2001)				
Medio ¹	Fuente ²	Dimensión ³	Año	Autores y Título
ESP	ICSM	PRO	2001	Harry M. Sneed; Impact Analysis of Maintenance Tasks for a Distributed Object-oriented System.
ESP	ICSM	PRO	2001	James S. O'Neal, Doris L. Carver; Analyzing the Impact of Changing Requirements.
ESP	ICSM	PRO	2001	Rob J. Kusters, Fred J. Heemstra; Software maintenance: an approach towards control.
ESP	ICSM	RIE	2001	Norman F. Schneidewind; Investigation of the Risk to Software Reliability and Maintainability of Requirements Changes.
ESP	CSMR	GES	1999	M. Polo, M. Piattini, F. Ruiz and C. Calero; MANTEMA: A Complete Rigorous Methodology for Supporting Maintenance based on the ISO/IEC 12207 Standard.
ESP	CSMR	HUM	2001	M. Di Penta, G. Casazza, G. Antoniol, and E. Merlo; Modeling Web Maintenance Centers through Queue Models.
ESP	CSMR	MED	1999	W. Lam, M. Loomes and V. Shankararaman; Managing Requirements Change using Metrics and Action Planning.
ESP	CSMR	MEJ	1998	F. Niessink, H. van Vliet; Towards Mature Measurement Programs.
ESP	CSMR	MOD	1998	S. Stoecklin, D. Williams, P. Stoecklin; Tailoring the Process Model for Maintenance and Reengineering.
ESP	CSMR	MOD	2000	A. Tomer and S. Schach; The Evolution Tree: A Maintenance-Oriented Software Development Model.
ESP	CSMR	MOD	2001	M. Kajko-Mattsson, U. Westblom, S. Forssander, G. Andersson, M. Medin, S. Ebarasi, T. Fahlgren, S.-E. Johansson, S. Törnquist, and M. Holmgren; Taxonomy of Problem Management Activities.
ESP	CSMR	PRO	1999	P. Bengtsson and J. Bosch; Architecture Level Prediction of Software Maintenance.
ESP	JSME	CAL	2000	Frank Niessink, Hans van Vliet; Software maintenance from a service perspective.
ESP	JSME	HUM	1998	Wui-Gee Tan, Guy G. Gable; Attitudes of maintenance personnel towards maintenance work: a comparative analysis.
ESP	JSME	INF	1998	Mark T. Dishaw, Diane M. Strong; Assessing software maintenance tool utilization using task-technology fit and fitness-for-use models.
ESP	JSME	INF	2001	Macario Polo Usaola, Mario Piattini Velthuis, Francisco Ruiz González; MANTOOL: a tool for supporting the software maintenance process.
ESP	JSME	MEJ	1998	Lionel Briand, Yong-Mi Kim, Walcélio Melo, Carolyn Seaman, Victor R. Basili; Q-MOPP: qualitative evaluation of maintenance organizations, processes and products.
ESP	JSME	MOD	1999	Barbara A. Kitchenham, Guilherme H. Travassos, Anneliese von Mayrhauser, Frank Niessink, Norman F. Schneidewind, Janice Singer, Shingo Takada, Risto Vehvilainen, Hongji Yang; Towards an ontology of software maintenance.
BD	APSEC	REU	1999	Oh-Cheon Kwon, Gyu-Sang Shin, Cornelia Boldyreff, Malcolm Munro; Maintenance with Reuse: An Integrated Approach Based on Software Configuration Management.
BD	ICSE	MEJ	2001	Mira Kajko-Mattsson, Stefan Forssander, Ulf Olsson; Corrective maintenance maturity model (CM3): maintainer's education and training.
BD	ICSE	MOD	1998	Mira Kajko-Mattsson; A conceptual model of software maintenance.

Publicaciones Relacionadas con la Gestión del Mantenimiento del Software (1998-2001)				
Medio ¹	Fuente ²	Dimensión ³	Año	Autores y Título
BD	ISESS	GES	1999	Macario Polo, Mario Piattini, Francisco Ruiz, Coral Calero; MANTEMA: A Software Maintenance Methodology Based on the ISO/IEC 12207 Standard.
BD	IWPC	INF	1999	Alex Sellink, Chris Verhoef; An Architecture for Automated Software Maintenance.
BD	SEN	HUM	1999	Macario Polo, Mario Piattini, Francisco Ruiz, Coral Calero; Roles in the Maintenance Process.
BD	SOFT	HUM	2000	Ramkumar Ramaswamy; How to Staff Business-Critical Maintenance Projects.
BD	SOFT	MED	1998	Norman F. Schneidewind; How To Evaluate Legacy System Maintenance.
BD	TSE	MED	1999	Norman F. Schneidewind; Measuring and Evaluating Maintenance Process Using Reliability, Risk, and Test Metrics.
LNCS	PROFES	INF	1999	K.Chiriatti; An Experience in Improvement of Maintenance Process Using CASE Tools.
LNCS	PROFES	MEJ	2001	Macario Polo, Mario Piattini, Francisco Ruiz, Mar Jiménez; Assessment of Maintenance Maturity in IT Departments of Public Entities: Two Case Studies.
KLU	ASE	CAL	2000	Norman F. Schneidewind; Software quality control and prediction model for maintenance.
OTRAS	ICEIS	MOD	1999	Francisco Ruiz, Mario Piattini, Macario Polo, Coral Calero; Maintenance Types in the MANTEMA Methodology.
OTRAS	JSS	CAL	1999	Giuseppe Visaggio; Assessing the maintenance process through replicated, controlled experiments.
OTRAS	JSS	HUM	2000	Sam Ramanujan and Richard W. Scamell and Jaymeen R. Shah; An experimental investigation of the impact of individual, program, and organizational characteristics on software maintenance effort.
OTRAS	JSS	INF	1998	Mark T. Dishaw and Diane M. Strong; Supporting software maintenance with software engineering tools: A Computed task-technology fit analysis.

Tabla A-1. Lista de publicaciones relacionadas con la Gestión del Proceso de Mantenimiento.

Las siglas y abreviaturas utilizadas tienen el siguiente significado:

1) Medios:

- BD => Revistas y conferencias publicadas por ACM y/o IEEE-CS, incluidas en las respectivas bibliotecas digitales.
- ESP => Revistas y conferencias especializadas en el mantenimiento de software.
- KLU => Revistas publicadas por la editorial Kluwer.
- LNCS => Conferencias publicadas en la colección “*Lecture Notes in Computer Science*” de la editorial Springer-Verlag.
- OTRAS => Otras revistas y conferencias consultadas a través de varios buscadores web de bibliografía científica.

2) Fuentes:

- APSEC => “*Asia Pacific Software Engineering Conference*”.
- ASE => “*Annals of Software Engineering*”.
- CSMR => “*European Conference on Software Maintenance and Reengineering*”.
- ICEIS => “*International Conference on Enterprise Information Systems*”.
- ICSE => “*International Conference on Software Engineering*”.
- ICSM => “*International Conference on Software Maintenance*”.
- ISESS => IEEE “*International Software Engineering Standards Symposium*”.
- IWPC => “*International Workshop on Program Comprehension*”.
- JSME => “*Journal of Software Maintenance and Evolution*” (hasta el año 2000 se llamaba “*Journal of Software Maintenance – Research and Practice*”).
- JSS => “*Journal of Systems and Software*”.
- PROFES => “*International Conference on Product Focused Software Process Improvement*”.
- SEN => ACM SIGSOFT “*Software Engineering Notes*”.
- SOFT => IEEE “*Software*”.
- TSE => IEEE “*Transactions on Software Engineering*”.

3) Dimensiones:

- ALI => alineamiento organizacional.
- CAL => gestión de la calidad.
- GES => gestión del PMS (en general).
- HUM => gestión de recursos humanos.
- INF => infraestructura (herramientas y entornos).
- MED => proceso de medida, métricas.
- MEJ => mejora y evaluación del proceso.
- MOD => modelos de proceso, métodos y ciclos de vida para el PMS.
- PRO => gestión del proyecto (planificación, seguimiento y control).
- REU => gestión de la reutilización.
- RIE => gestión de riesgos.

B. Metodología MANTEMA.

En este anexo se presenta la visión del PMS que ofrece MANTEMA, una metodología para mantenimiento de software que integra todas las actividades relacionadas con este proceso. El objetivo de MANTEMA es convertir el mantenimiento de software en un proceso controlable y mensurable mediante la identificación y definición clara de todos los elementos (software, documentos, personas, tareas...) que intervienen en este proceso. MANTEMA ha sido desarrollada por el grupo Alarcos del Departamento de Informática de la Universidad de Castilla La Mancha, en colaboración con la empresa Atos ODS. Para su estudio en profundidad está publicado un informe técnico (Polo et al, 1999a). También se puede consultar la tesis doctoral de Macario Polo (Polo, 2000). Otras publicaciones del grupo Alarcos referidas a esta metodología son:

- a) (Ruiz et al, 1998); donde se introduce la primera propuesta genérica de la metodología, haciendo hincapié en que está basada en el modelo de ciclo de vida del software de ISO 12207, y en la inclusión del objetivo de mantenibilidad.
- b) (Polo et al, 1999b); donde se presenta el modelo organizacional utilizado, que está basado en la definición de tres tipos de organizaciones (cliente, mantenedor y usuario) y en diferentes roles en cada una de ellas.
- c) (Ruiz et al, 1999c); donde se presentan los cinco tipos de mantenimiento contemplados en MANTEMA, así como la estructura general en grupos de actividades comentada en el anexo B, y el método para determinar el tipo de mantenimiento cuando se recibe una petición de modificación.
- d) (Polo et al, 1999c) y (Piattini et al, 2000); que presentan la estructura general de actividades y tareas de las versiones 1 y 2, respectivamente (ver apartado B.2).
- e) (Polo et al, 1999d), (Piattini et al, 1999) y (Polo et al, 1999e); que explican cómo se realizó la adaptación de la norma ISO 12207 para su utilización en MANTEMA.
- f) (Polo et al, 2001e); donde se presentan algunos cambios sobre la versión 2 en la estructura de actividades y tareas, basados en las primeras experiencias reales.

B.1. Características Generales.

MANTEMA aborda por entero el proceso de mantenimiento, intentando disminuir los costes de todas sus actividades. Para conseguirlo, una de las primeras tareas que debe completarse es la definición clara del conjunto de actividades que se deben realizar a lo largo del PMS. En este sentido, se han seguido las recomendaciones de ISO 12207 (ISO/IEC, 1995), IEEE 1219 (IEEE, 1993) y Pressman (1998), que abogan por ejecutar cada petición de modificación según el tipo de mantenimiento que le corresponda. Por ello, en MANTEMA se definen cinco tipos diferentes de mantenimiento (Ruiz et al, 1999c), cada uno con un conjunto diferente de tareas:

1. *Correctivo urgente*, que tiene lugar cuando existe un error en el software que bloquea el funcionamiento normal de la organización o de la aplicación, siendo crítico el tiempo de solución.

2. *Correctivo no urgente*, que ocurre cuando existe un error en el software que no es crítico, pero que tal vez impida el funcionamiento de la aplicación o el normal funcionamiento de la empresa en un periodo de tiempo relativamente corto.
3. *Perfectivo*, que tiene lugar cuando se van a añadir nuevas características o funcionalidades al software en explotación.
4. *Adaptativo*, que se aplica cuando el software en explotación va a modificarse para que continúe funcionando correctamente en un entorno que ha cambiado.
5. *Preventivo*, que es aplicado cuando se desean mejorar las características internas de un producto software; por ejemplo, para hacerlo más fácilmente mantenible.

Sin embargo, durante la construcción y aplicación de la metodología se observó que, a pesar de que las intervenciones de correctivo no urgente, perfectivo, preventivo y adaptativo poseen características propias que las diferencian unas de otras, sus líneas de diseño y ejecución son bastante similares por lo que, en la versión 2 de MANTEMA, se decidió agrupar estos cuatro tipos de mantenimiento bajo una única denominación, “mantenimiento planificable”, pasando a denominar “no planificable” al correctivo urgente.

Además de detallar la secuencia de operaciones que se debe ejecutar para las intervenciones de cada tipo de mantenimiento, MANTEMA hace una especial consideración a la puesta en marcha del proceso de mantenimiento, identificando y definiendo dos conjuntos adicionales de actividades:

- En el primero, “*Actividades y tareas iniciales comunes*”, se engloban todas las actividades que deben ser ejecutadas al comenzar el proceso de mantenimiento y que serán anteriores a la ejecución de cualquier intervención.
- El segundo, “*Actividades y tareas finales comunes*”, agrupa las actividades que serán realizadas tanto al finalizar el proceso de mantenimiento como aquellas que serán ejecutadas con posterioridad a las intervenciones, independientemente de su tipo.

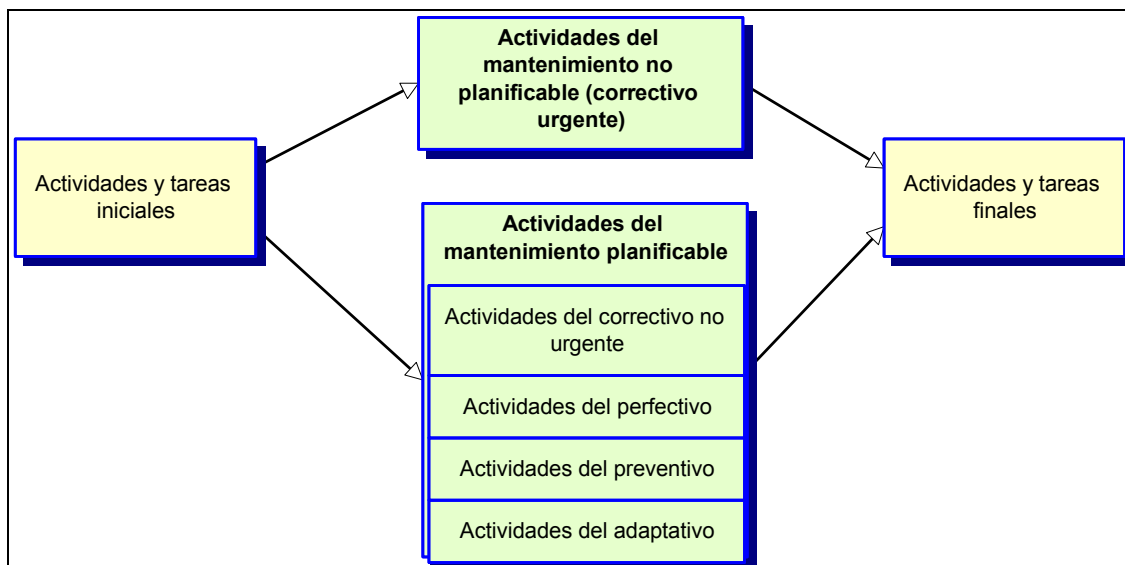


Figura B-1. Vista general de la estructura de actividades de MANTEMA.

Con este planteamiento, el PMS definido por MANTEMA puede entenderse como el grafo polietápico que se muestra en la Figura B-1. Siguiendo la idea de ISO 12207, cada nodo del grafo está formado por un conjunto de actividades que, a su vez, están formadas por un conjunto de tareas. Por tanto, en MANTEMA se establece una jerarquía de descomposición de trabajos de 4 niveles: proceso, grupo de actividades, actividad, y tarea. Cada tarea se define especificando los siguientes elementos:

- Entradas, que son los elementos necesarios para la correcta realización de la tarea. Estos elementos podrán ser programas, documentos, etc., y ser tomados bien de tareas anteriores, bien del entorno del proceso.
- Salidas, que son los elementos que se generan tras la realización de la tarea. Las salidas podrán ir dirigidas a otras tareas posteriores o bien al entorno.
- Técnicas, que son las técnicas que pueden utilizarse para realizar la tarea.
- Métricas, que deben recogerse para mantener el proceso y el producto bajo control.
- Responsables, representados por los roles encargados de la realización de la tarea, y que serán algunos de los enumerados en (Polo et al., 1999b).
- Interfaces con otros procesos, que se establecerán durante la realización de la tarea con el resto de procesos definidos por la organización para el ciclo de vida software, por ejemplo, con el proceso de “Gestión de la Configuración”.

Al hilo de este último elemento, en el PMS definido en la metodología se integran las actividades necesarias para que algunos de los procesos de soporte definidos en ISO 12207 ayuden a llevar a cabo el mantenimiento. De este modo, la relación del proceso de mantenimiento descrito en MANTEMA con el resto de procesos de soporte se puede representar con la Figura B-2, que muestra como procesos satélites al mantenimiento aquellos que no se han integrado y que son con los que se establece interfaz en algún momento.

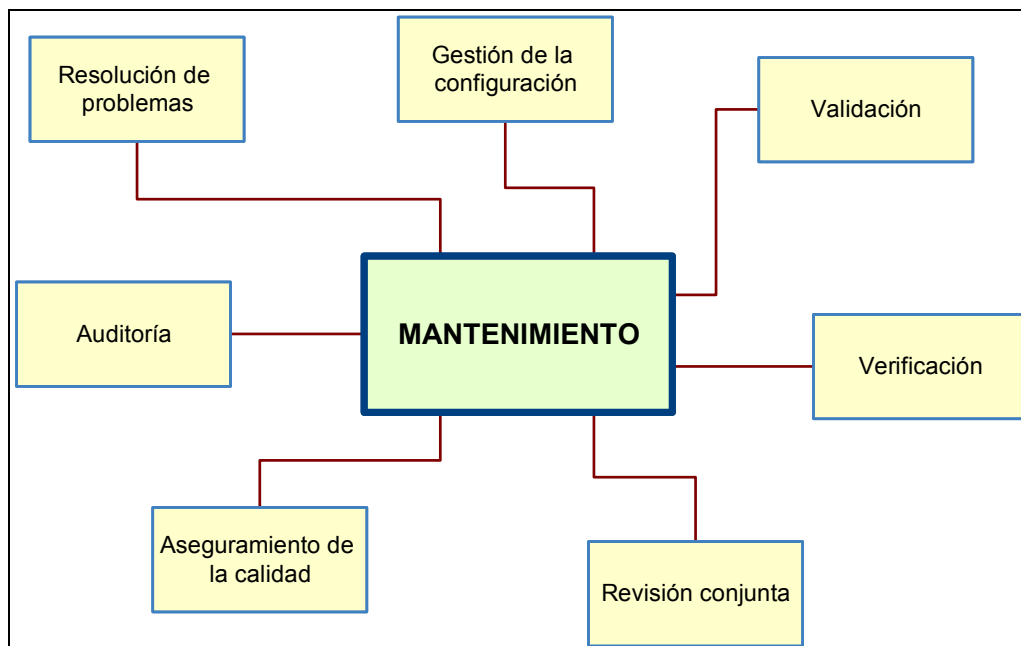


Figura B-2. Procesos de soporte con los que se establece un interfaz.

B.1.1. Participantes.

Se han definido tres organizaciones participantes en el proceso de mantenimiento (Polo et al., 1999b). Dependiendo de la situación, cada una de estas organizaciones puede ser una real diferente, o coincidir varias en una sola real. Además, dentro de cada una de dichas organizaciones, se han distinguido diversos perfiles o roles.

Las tres organizaciones virtuales y sus respectivos perfiles son las siguientes:

- a) Cliente: es la organización propietaria del software; y por tanto, la que recibe el servicio de mantenimiento. Sus perfiles son:
 - a.1) *Solicitante*: es quien presenta las solicitudes de modificación, establece los requerimientos necesarios para su implementación, y los entrega al mantenedor.
 - a.2) *Organización del sistema*: es el departamento que conoce el sistema que será mantenido.
 - a.3) *Atención a usuarios*: es el departamento que presta asistencia a los usuarios.
- b) Mantenedor: es la organización que realiza el servicio de mantenimiento. Sus perfiles son:
 - b.1) *Gestor de peticiones*: acepta o rechaza las peticiones modificación y decide el tipo de mantenimiento que debe aplicarse.
 - b.2) *Planificador*: planifica la cola de peticiones de modificación aceptadas.
 - b.3) *Equipo de mantenimiento*: es el grupo de personas que implementa las solicitudes de modificación.
 - b.4) *Responsable de mantenimiento*: prepara el proceso de mantenimiento y establece las normas y procedimientos necesarios para utilizar la metodología de mantenimiento (en este caso, MANTEMA).
- c) Usuario: es la organización que utiliza el software objeto del mantenimiento. Su único perfil es:
 - c.1) *Usuario*: utiliza el software mantenido y comunica las incidencias a “Atención a usuarios”.

B.2. Estructurada Detallada de Actividades y Tareas.

En esta sección se describen con mayor detalle los nodos de la Figura B-1.

B.2.1. Actividades y Tareas Iniciales Comunes.

Este grupo consta de tres actividades. En la realización de las dos primeras (Estudio inicial y Planificación del proceso) se prepara el proceso de mantenimiento, mientras que la tercera está dedicada a la recepción y clasificación de las peticiones de modificación.

Durante la primera actividad (*Estudio inicial*), la organización de mantenimiento recoge información acerca del software que se va a mantener, con el objeto de realizar la planificación del proceso de mantenimiento y, en su caso, de presentar una propuesta de mantenimiento a la organización Cliente.

Una vez que la información anterior está recopilada, las dos siguientes tareas están dedicadas a la preparación de la propuesta de mantenimiento y a la redacción y firma del contrato de prestación del servicio, ambas muy importantes en el caso de que exista una relación de externalización (*outsourcing*) entre la organización que realiza el mantenimiento y la organización cliente.

La segunda actividad es la Planificación del proceso de mantenimiento. Durante esta actividad, la organización de mantenimiento adquiere conocimiento de la aplicación. En los casos en que existe externalización, la organización de mantenimiento actúa durante esta tarea simplemente observando cómo ejecuta sus tareas el actual equipo de mantenimiento. Tras este periodo “mudo”, la organización de mantenimiento debe entregar al cliente documentación completa del software que incluya informes de auditoría, posibles mejoras, etc., además de ir construyendo documentación de consumo interno que incluya valores de métricas, tablas de referencias cruzadas, diccionarios de datos, etc. Dentro todavía de la segunda actividad, también se definen los procedimientos que deberán seguirse para presentar las peticiones de modificación, se implementa el proceso de gestión de configuración (en caso de que se carezca de uno) y, a ser posible, se preparan los entornos en que se realizarán las pruebas.

A partir del momento en que queda completada la segunda actividad, la organización de mantenimiento está preparada para ejecutar las acciones necesarias para servir las peticiones de modificación. Entraríamos, por tanto, en un conjunto de actividades y tareas cíclico, en el sentido de que serán ejecutadas para cada Petición de Modificación (PM) que se reciba.

Precisamente la tercera y última actividad de este conjunto inicial de actividades y tareas está dedicada a la Recepción de las PM's y a su clasificación según uno de los dos tipos de mantenimiento que distinguimos en la sección anterior (aunque, para las peticiones de planificable, será necesario seguir precisando si se trata de correctivo no urgente, perfectivo, preventivo o adaptativo, ya que la forma de actuar difiere ligeramente en cada caso).

B.2.2. Actividades y Tareas del Mantenimiento no Planificable.

En la Tabla B-1 se detalla la serie de actividades y tareas que deben seguir las intervenciones de mantenimiento que, en la tercera actividad del nodo inicial del proceso, se hayan clasificado como correspondientes a mantenimiento “no planificable”, es decir, correctivo urgente.

Actividad =>	NP1 Análisis del error	NP2 Intervención correctiva urgente			NP3 Cierre intervención
Tarea =>	NP1.1 Investigar y analizar causas	NP2.1 Realizar acciones correctivas	NP2.2 Cumplimentar documentación	NP2.3 Ejecutar pruebas unitarias	NP3.1 Pasar a producción
Entradas	Producto software en explotación con error bloqueante o crítico. PM	Conjunto de elementos software a corregir.	Elementos software antiguos (con errores visibles). Elementos software corregidos.	Elementos software corregidos. Casos de prueba.	Elementos software corregidos y probados.

Actividad =>	NP1 Análisis del error	NP2 Intervención correctiva urgente			NP3 Cierre intervención
Tarea =>	NP1.1 Investigar y analizar causas	NP2.1 Realizar acciones correctivas	NP2.2 Cumplimentar documentación	NP2.3 Ejecutar pruebas unitarias	NP3.1 Pasar a producción
Salidas	Conjunto de elementos software a corregir.	Conjunto de elementos software corregidos.	Documentación de las acciones correctivas realizadas.	Elementos software corregidos y probados. Documentación con las pruebas unitarias realizadas.	Producto software en explotación corregido.
Técnicas		Codificación.		Técnicas de prueba del software.	
Responsable	Equipo de mantenimiento. Usuario.	Equipo de mantenimiento.	Equipo de mantenimiento.	Equipo de mantenimiento.	Equipo de mantenimiento. Usuario.
Interfaces con otros procesos		Aseguramiento de la calidad. Gestión de la configuración.		Aseguramiento de la calidad.	Gestión de la configuración.

Tabla B-1. Estructura del mantenimiento no planificable.

B.2.3. Actividades y Tareas del Mantenimiento Planificable.

En este tipo de mantenimiento se incluye la definición de las actividades y tareas para los mantenimientos correctivo no urgente, perfectivo, preventivo y adaptativo. Cada nodo de la Figura B-3 representa una tarea de estos tipos de mantenimiento. Los nodos de dicha figura se pueden detallar utilizando tablas del estilo de las usadas para el no planificable, de manera que también para estas tareas se especifican entradas, salidas, técnicas, etc. No se incluyen en este documento debido a su gran extensión.

Como se puede observar en la citada figura, la secuencia de tareas seguida por las intervenciones correctivas no urgentes y perfectivas es exactamente la misma, mientras que existen algunas diferencias entre éstas y las preventivas y adaptativas.

B.2.4. Actividades y Tareas Finales Comunes.

MANTEMA también detalla este último nodo de la Figura B-1 mediante tablas similares a las usadas anteriormente. Este nodo consta de las siguientes cuatro actividades:

- Registro de la intervención, tras el cual la intervención (incluyendo toda su documentación asociada) queda registrada según los procedimientos que se establecieron en la actividad “Planificación del proceso”.
- Actualización de la base de datos histórica, que consiste en almacenar (si no se ha hecho ya) los valores de las diferentes métricas que deben recogerse en cada tarea.
- Retirada, que será realizada conforme a la Retirada indicada en ISO 12207.
- Fin de la externalización, que ocurrirá sólo si ha existido relación de “outsourcing” entre la organización que hace el mantenimiento y el cliente.

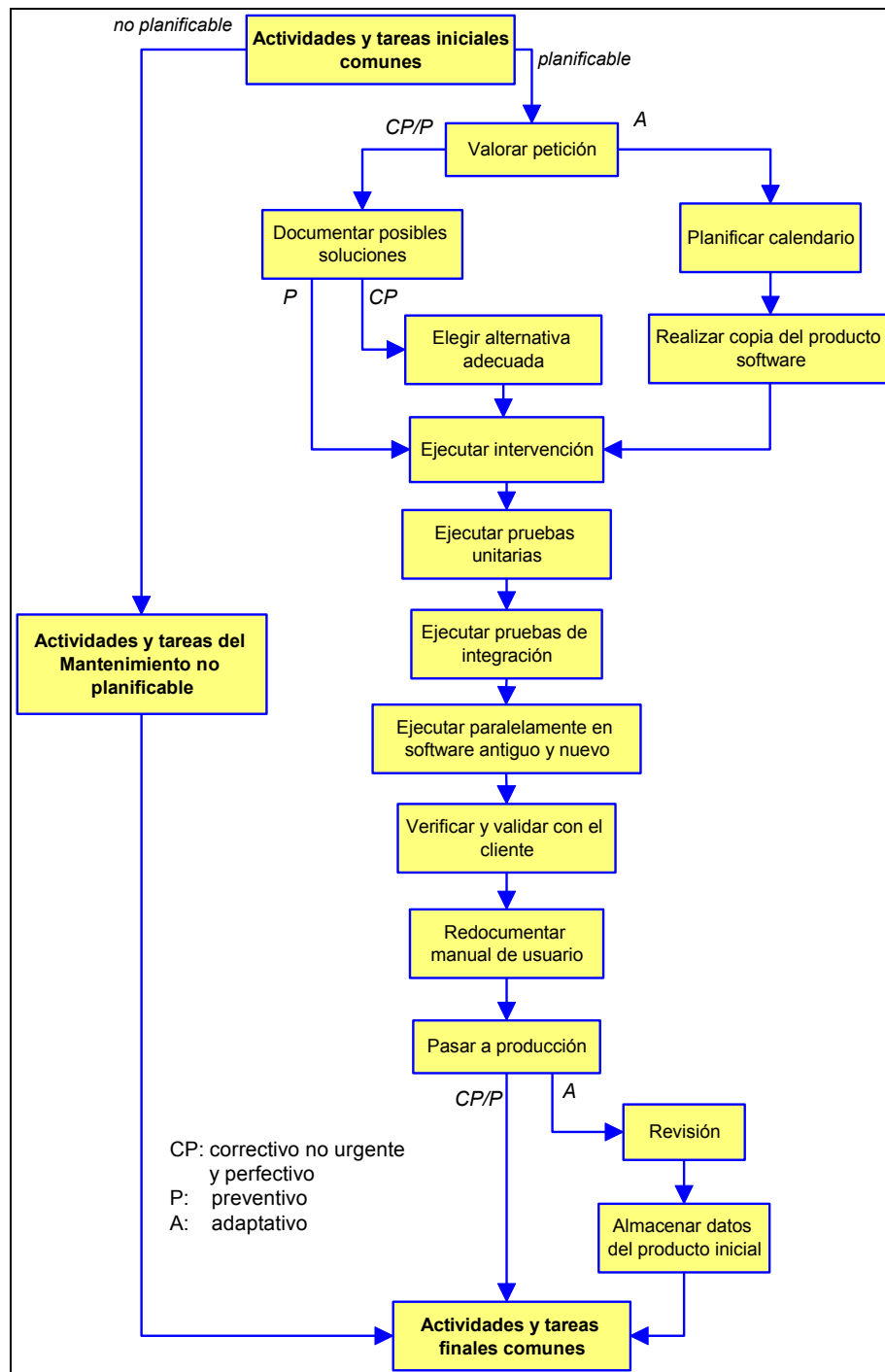


Figura B-3. Estructura del mantenimiento planificable.

B.2.5. Documentación.

En MANTEMA se concede mucha importancia a la documentación. De hecho, gran parte de los elementos que las tareas utilizan como entradas o salidas son documentos que se generan durante el proceso. En la metodología se definen plantillas para prácticamente todos los tipos de documentos que se pueden generar durante el mantenimiento. En la Tabla B-2 se recoge una lista con algunos de ellos.

MANTEMA: Plantillas de documentos
1. Cuestionario inicial
2. Propuesta de mantenimiento
3. Contrato de mantenimiento
4. Tabla de factores de riesgo
5. Resumen técnico
6. Petición de modificación
7. Acciones correctivas realizadas
8. Pruebas unitarias realizadas
9. Diagnóstico y posibles soluciones
10. Alternativas de implementación
11. Acciones perfectivas realizadas
12. Lista de elementos software y propiedades mejorables
13. Acciones preventivas realizadas
14. Plan de migración
15. Notificación de futura migración
16. Medidas del producto
17. Plan de mantenimiento del periodo

Tabla B-2. Lista de plantillas de documentos incluidas en MANTEMA.

C. MOF: Meta-Object Facility.

MOF (*Meta-Object Facility*) es una norma aprobada por el OMG (*Object Management Group*) para la definición, representación y gestión de metadatos (OMG, 2002a). La versión de MOF utilizada en esta tesis ha sido la 1.3, publicada en marzo de 2000, aunque la versión más actual de MOF es la 1.4, que ha sido publicada en abril de 2002. MOF y sus especificaciones asociadas incluyen dos partes principales:

- La arquitectura MOF de 4 niveles de metadatos, que proporciona un patrón genérico para la construcción de sistemas centrados en los metadatos (apartado C.1).
- El Modelo MOF, que es el lenguaje estándar y abstracto para los metamodelos que definen diferentes clases de metadatos (apartado C.2).

Un aspecto importante a considerar con respecto al estándar MOF es su concepto de lo que es un modelo. Habitualmente, el término modelo es usado para describir algo en el mundo real. En este sentido, un modelo sería una abstracción de la realidad y dependería del punto de vista del modelador. Toda persona que tenga que construir o entender un sistema complejo, necesita un modelo del mismo en el que tendrán que estar incluidos todos los metadatos del sistema, aunque en ciertas ocasiones el interés pueda estar centrado sólo en ciertos componentes (programa A o programa B) o en ciertos grados de detalle (diagrama de conexiones) del sistema. En el contexto de MOF, un modelo no lo es en el sentido usual de la palabra, ya que no tiene porqué describir necesariamente algo del mundo real y no tiene porqué definir cosas interesantes para los modeladores. En MOF este término tiene un significado más amplio: Un modelo es cualquier colección de metadatos que están relacionados de la siguiente manera:

- Los metadatos describen información interrelacionada de alguna forma.
- Todos los metadatos definidos deben estar en concordancia con una serie de reglas que definen su estructura y consistencia (están basados en un lenguaje abstracto).
- Los metadatos tienen significado en un marco semántico común.

C.1. Arquitectura Conceptual.

El aspecto central de la aproximación MOF es soportar cualquier clase de metadatos y permitir la incorporación de otros nuevos cuando sea necesario. Para lograr este objetivo MOF utiliza una arquitectura de cuatro capas similar a otras propuestas de estándares para metadatos como OIM (*Open Information Model*) de la *Meta-Data Coalition* (MDC, 1999) - que recientemente se ha integrado en OMG y ha adoptado MOF - y CDIF (*CASE Data Interchange Format*) de ISO/IEC (2000c). En general, todos estos modelos incluyen las cuatro capas siguientes:

- La capa de *objetos de usuario* engloba la información que queremos describir. Esta información suele conocerse como “datos”.
- La capa de *modelos* comprende los metadatos que describen información. Los metadatos se agregan formando modelos (según la definición comentada anteriormente).

- La capa de *metamodelos* incluye las descripciones (meta-metadatos) que definen la estructura y semántica de los metadatos. Meta-metadatos se agregan formando metamodelos. Un metamodelo también puede considerarse como un lenguaje para describir diferentes clases de datos.
- La capa del *meta-metamodelo* incluye la descripción de la estructura y semántica de los meta-metadatos. Por tanto, es un lenguaje para definir diferentes clases de metadatos.

Las principales ventajas de esta arquitectura son las siguientes:

- a) Si el meta-metamodelo es suficientemente rico, será posible soportar gran parte de los tipos de meta-información imaginables.
- b) Se pueden asociar diferentes clases de metadatos.
- c) Si se utiliza el mismo meta-metamodelo, será posible intercambiar tanto metadatos (modelos) como meta-metadatos (metamodelos).

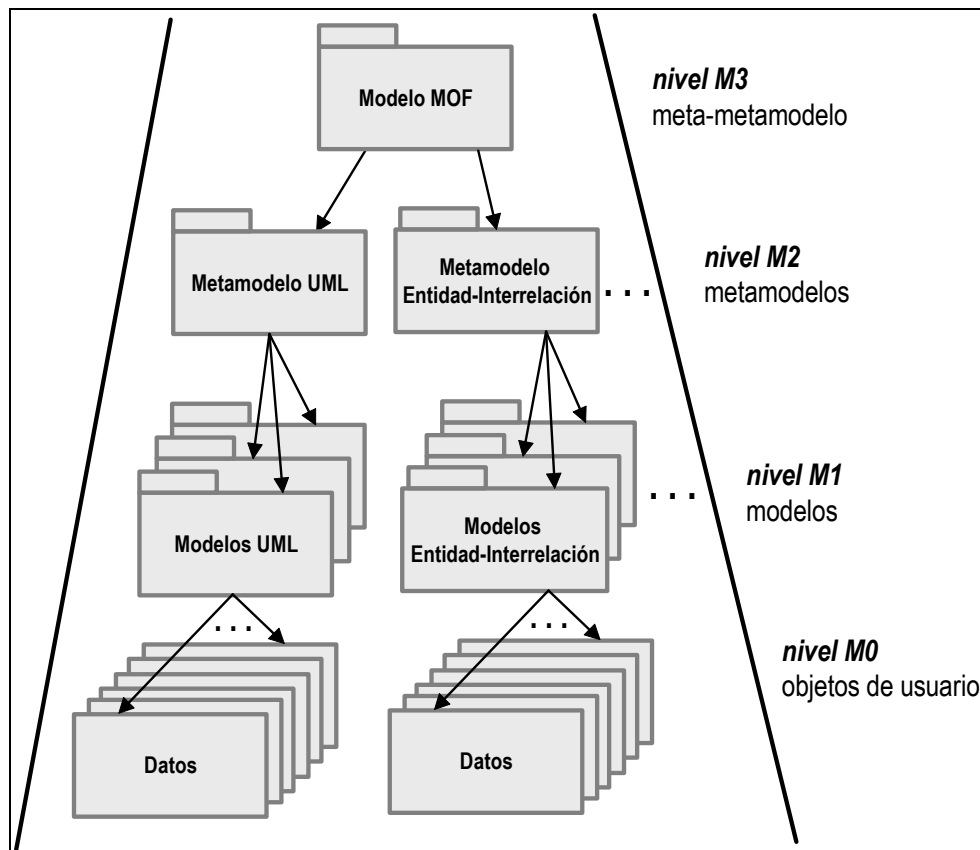


Figura C-1. Arquitectura de cuatro niveles para metadatos de MOF.

La arquitectura de cuatro niveles de MOF (mostrada en la Figura C-1 con ejemplos de instancias de UML y entidad-interrelación) está basada en la propuesta general anterior, pero con algunas particularidades que la hacen más potente y flexible que otras propuestas. Estas características distintivas son:

- MOF es orientado a objetos, soportando constructores de metamodelado que están alineados (aunque son algo más simples) con los constructores de UML.

- El meta-metamodelo (es decir, el Modelo MOF) es auto-descriptivo. Esto significa que el Modelo MOF es definido formalmente usando sus propios constructores de metamodelado. Por esta razón, en la Figura C-1 se ha representado el Modelo MOF usando el símbolo de paquete (igual que en UML).

MOF busca soportar cualquier clase de metadatos que pueda ser descrita usando técnicas de modelado orientado a objetos. Teniendo en cuenta este aspecto, el marco de trabajo establecido por MOF debe poder soportar diferentes metamodelos (en el nivel M2). El nivel M3, con el Modelo MOF que se presenta en el apartado C.2, es necesario precisamente para dar soporte a dicha integración de diferentes metamodelos.

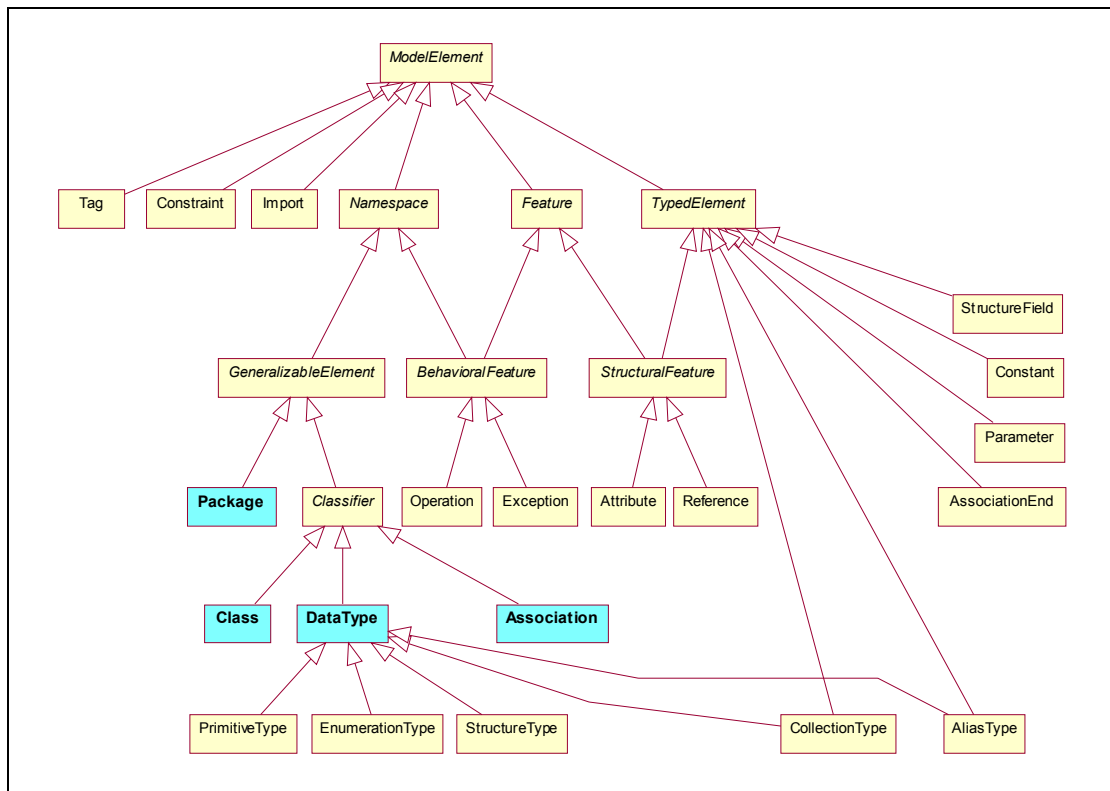


Figura C-2. Jerarquía de herencia de las clases del Modelo MOF.

C.2. El Modelo MOF.

El "*Modelo MOF*" es el meta-metamodelo (nivel M3 de la arquitectura conceptual de MOF) para la descripción de metamodelos del nivel M2. Como se ha presentado anteriormente, es un lenguaje abstracto para definir metamodelos en MOF. Es decir, desempeña un papel análogo al de UML (OMG, 2001a) como lenguaje abstracto para la definición de modelos UML (que estarían en el nivel M1 de la arquitectura MOF). El Modelo MOF está alineado con UML en el sentido de que ambos comparten los conceptos y constructores centrales de modelado. De esta forma, conceptos de UML tales como Clase, Asociación, Paquete, Tipo de Dato, Atributo, Operación, Restricción, etc., tienen sus equivalentes en MOF, aunque con algunas simplificaciones. La diferencia entre ambos reside, por un lado, en que MOF supone un nivel de abstracción más que UML y, por otro lado, en que MOF y UML han sido diseñados para dos

clases distintas de modelado (metadatos en el caso de MOF y datos, es decir objetos, en el caso de UML). En la Figura C-2 se muestra la jerarquía de herencias de las clases (en realidad, metaclasses) del Modelo MOF. Hemos dejado los nombres originales en inglés para facilitar su identificación al consultar el estándar.

C.2.1. Constructores para Metamodelado.

El lenguaje abstracto de MOF para definir metamodelos está formado por constructores de metamodelado. Los cuatro *constructores principales de metamodelado* proporcionados por MOF son Clase, Asociación, Paquete y Tipo de Dato (en la Figura C-2 se muestran resaltados). A continuación resumimos sus principales características.

C.2.1.1. Clases MOF.

El constructor Clase (*class*) es utilizado para definir el tipo de los nodos conceptuales de metadatos, es decir, tipos de meta-objetos. Las clases definidas en el nivel M2 tienen instancias en el nivel M1. Estas instancias tienen identidad como objetos, estado y comportamiento. En consecuencia, el estado y comportamiento de las instancias del nivel M1 están definidos por la clase del nivel M2. La colección de todas las instancias de una clase forma la “extensión” de la clase. Las principales propiedades de una clase MOF son: nombre, lista de atributos, lista de supertipos (las clases de las que hereda propiedades, es decir, atributos, referencias y operaciones), abstracción (un indicador de si la clase es abstracta, es decir, si no puede tener instancias; estas clases se crean con el fin de que otras clases puedan heredar de ellas), y *singleton* (que indica si la clase sólo puede tener una única instancia). Está permitida la herencia múltiple entre clases al estilo de CORBA.

Una Clase puede tener tres tipos de características estructurales: Atributos, Operaciones y Referencias. También puede contener definiciones de Excepciones, Constantes, Tipos de Datos, Restricciones y otros elementos.

C.2.1.1.1. Atributos.

Los Atributos son características estructurales que definen contenedores de valores que pertenecen a instancias de la clase *Atributo*. Cada atributo tiene las siguientes propiedades importantes: nombre, tipo (una Clase o un Tipo de Dato), multiplicidad (cuantos valores están permitidos para cada instancia o clase, si se puede repetir el mismo valor en varias instancias, y si los valores están ordenados), alcance (que puede ser a nivel de instancia o a nivel de clase, es decir, un valor para cada instancia o uno sólo para toda la clase), derivabilidad (si el valor del atributo se almacena como parte del estado del objeto o es calculado a partir de otras informaciones), y cambiabilidad (si el valor puede ser modificado directamente).

La agregación de atributos se realiza asignando una Clase (que incluye a su vez una colección de atributos) al tipo del atributo. Se dice entonces que dicho atributo es “compuesto” (*composite*). En cambio, un atributo no-agregado (*non-aggregate*) es aquel que tiene asignado un Tipo de Dato.

C.2.1.1.2. Operaciones.

Las Operaciones (*operations*) también son propiedades de las Clases, utilizadas para acceder a su comportamiento asociado. Las operaciones no especifican el comportamiento o los métodos que lo implementan, sino que tan solo especifican los nombres y tipos necesarios para invocar dicho comportamiento. Las Operaciones tienen las siguientes propiedades: nombre, lista posicional de Parámetros, tipo del resultado, y lista de las Excepciones que pueden ser invocadas.

Un *Parámetro* MOF tiene un nombre, un tipo (una Clase o un Tipo de Dato, una dirección (“in”, “out” o “in out” según el valor se pase del cliente al servidor, del servidor al cliente o en ambas direcciones, respectivamente), y multiplicidad (definida igual que para los Atributos).

C.2.1.1.3. Referencias.

El modelo MOF provee dos constructores para modelar interrelaciones entre clases: Asociaciones y Atributos. Las Asociaciones ofrecen un modelo computacional orientado a las consultas (porque proveen un objeto que encapsula una colección de enlaces sobre los que se pueden realizar consultas globales), mientras que los Atributos proveen un modelo computacional orientado a la navegación (habitualmente el usuario realiza operaciones sobre un atributo de una instancia de una clase). Las Referencias son una tercera forma de modelar que permite dotar de las ventajas navegacionales (de los atributos) a las Asociaciones. Una Referencia es una característica estructural de una Clase que permite que la clase “conozca” cuales de sus instancias están vinculadas con un determinado Fin de Asociación, es decir, en cuales Asociaciones participa la Clase. Además, las operaciones resultantes sobre la Referencia son similares a las definidas sobre Atributos con la misma multiplicidad. Una Referencia tiene el siguiente contenido: nombre, fin de asociación referenciada, y cambiabilidad.

C.2.1.2. Asociaciones MOF.

Asociación (*association*) es el constructor para definir relaciones binarias entre Clases. Las instancias conceptuales de una Asociación son “enlaces” binarios dirigidos que conectan un tipo de meta-objeto con otro. Una Asociación MOF del nivel M2 define interrelaciones (enlaces) del nivel M1 entre pares de instancias de Clases. Estos enlaces no tienen identidad como objetos y, por tanto, no pueden tener atributos u operaciones. Las asociaciones MOF tienen tres diferencias sustanciales con las de UML:

- sólo pueden ser binarias (sólo tienen dos Finales de Asociación), mientras que en UML pueden tener cualquier grado;
- sus instancias, es decir los enlaces, no tienen identidad de objetos y, por tanto al contrario que en UML, no pueden tener atributos y no pueden ser finales de otros enlaces; y
- sus enlaces son implícitamente únicos, es decir, no se pueden crear enlaces duplicados entre el mismo par de nodos de metadatos, cosa que sí es posible en UML.

C.2.1.2.1. Finales de Asociación.

Cada Asociación tiene dos Finales de Asociación describiendo los dos finales de los enlaces. Cada Fin de Asociación tiene las siguientes propiedades: nombre, tipo (una Clase), multiplicidad, especificación de agregación, navegabilidad (qué controles de navegación pueden definirse) y cambiabilidad (si puede ser modificado). La multiplicidad de un Fin de Asociación es similar a la de Atributos y Operaciones, pero con las siguientes diferencias:

- la multiplicidad no se refiere al conjunto de enlaces sino al número mínimo y máximo de instancias de clase del otro Fin de Asociación que se pueden enlazar con cada instancia de clase de este Fin de Asociación (es una definición idéntica a la utilizada por algunos autores para las cardinalidades máxima y mínima en el modelo Entidad-Interrelación Extendido); y
- no está permitida la duplicidad de enlaces entre el mismo par de instancias de clase, es decir, la propiedad de unicidad siempre es cierta.

C.2.1.2.2. Semántica de las Asociaciones.

MOF permite dos tipos diferentes de interrelaciones entre instancias (es decir, mecanismos de agregación), llamados compuesto (*composite*) y no-agregado (*non-aggregate*). Una interrelación no-agregada es un vínculo débil entre instancias, que tiene las siguientes propiedades:

- no existen restricciones especiales sobre la multiplicidad de la interrelación;
- no existen restricciones especiales sobre el origen de las instancias en la interrelación; y
- la interrelación no influye sobre el ciclo de vida de las instancias relacionadas (la eliminación de una instancia no provoca la eliminación en cascada de las instancias relacionadas)

Por el contrario, una interrelación compuesta es un vínculo fuerte entre instancias, que tiene las siguientes propiedades:

- es asimétrica, habiendo un participante que desempeña el rol de “compuesto” y otro que desempeña el rol de “componente”, de forma que el segundo es parte del primero;
- una instancia no puede ser componente de más de un compuesto a la vez (en una misma interrelación);
- una instancia no puede ser componente de sí misma, sus componentes, los componentes de sus componentes, etc. (en una misma interrelación);
- cuando una instancia compuesta es eliminada, todos sus componentes, los componentes de sus componentes, etc. (para todas las interrelaciones compuestas en que participa) son también eliminados; y
- una instancia no puede ser componente de otra instancia de la extensión de un Paquete diferente (esto se conoce como “Regla de Cierre de Composición”).

Para asignar uno de los tipos de interrelaciones anteriores a una Asociación MOF se utiliza la propiedad de “agregación” de los Finales de Asociación. Así, una Asociación compuesta se especifica asignando los valores “verdadero” y “falso” a dicha propiedad para los

Finales de Asociación con los roles “compuesto” y “componente”, respectivamente. Además, en consonancia con las propiedades enumeradas antes, la multiplicidad del Final de Asociación “compuesto” deberá ser [0..1] o [1..1], para evitar que una instancia sea componente de varios compuestos dentro de la misma Asociación.

C.2.1.3. Paquetes MOF.

El Paquete (*package*) es el constructor del Modelo MOF que sirve para agrupar elementos dentro de un metamodelo. En el nivel M2, un Paquete provee una forma para particionar y modularizar el espacio del metamodelo. Los paquetes en M2 pueden contener muchas clases de elementos: otros Paquetes, Clases, Asociaciones, Tipos de Datos, Excepciones, Constantes, etc. En el nivel M1, las instancias de un Paquete actúan como el contenedor más externo de metadatos. Indirectamente, también son un mecanismo de definición de alcance para otros elementos del metamodelo, como Clases, Asociaciones y Tipos de Datos. Una instancia de un Paquete MOF (llamada un “objeto paquete”) es un objeto que gestiona una colección de metadatos relacionados de conformidad con el Paquete. La colección de metadatos gestionada por un objeto paquete recibe el nombre de “extensión” del paquete.

Los paquetes MOF difieren de los paquetes UML en que mientras UML permite dependencias cíclicas entre paquetes, en MOF no están permitidas; y en que los paquetes MOF tienen sus correspondientes definiciones conceptuales en el nivel de metadatos (M2), mientras que en UML no existen tales y sólo existen las instancias del nivel M1.

Además de ser un contenedor de Clases, Asociaciones y otros Paquetes (y también de definiciones de Tipos de Datos y Excepciones), un Paquete MOF también incluye las propiedades importaciones y supertipos, que tienen que ver con mecanismos para modularizar y reutilizar paquetes que se comentan en el siguiente apartado.

C.2.1.3.1. Mecanismos para Modularización y Reutilización.

El Modelo MOF provee los siguientes mecanismos para soportar la modularización y reutilización de metamodelos y componentes de metamodelos:

- a) **Generalización** (*generalization*): Los Paquetes pueden ser generalizados por (heredados desde) uno o más Paquetes de forma análoga a la herencia de clases. Cuando un Paquete hereda de otro, el primero (sub-Paquete) adquiere todos los elementos del segundo (Super-Paquete).
- b) **Anidamiento** (*nesting*): Un Paquete puede contener otros Paquetes, que a su vez pueden contener otros Paquetes, etc. Los Paquetes anidados no son directamente instanciables, ya que para existir una instancia de paquete anidado es necesaria la existencia de una instancia del Paquete contenedor, de la cual es un componente. La utilidad del anidamiento no es la reutilización sino la posibilidad de definir alcances a la parte deseada del contenido del paquete.
- c) **Importación** (*importing*): Es el mecanismo provisto por MOF para poder reutilizar algunos elementos de un metamodelo (o Paquete) y no otros. Un Paquete puede importar de uno o varios Paquetes. De esta manera, los componentes de los paquetes importados están disponibles para su uso en el paquete importador. Algunos ejemplos de la utilidad de la importación son: Atributos utilizando Tipos de datos importados, Operaciones que invocan Excepciones importadas, Clases cuyos supertipos son Clases importadas, Asociaciones que tienen algún Fin de Asociación que es una Clase importada, etc.

- d) **Agrupamiento** (*clustering*): Este mecanismo es un tipo especial de importación que vincula los Paquetes importador e importado en un grupo (*cluster*).

Mecanismos de Composición de Paquetes en MOF			
Constructor	Interrelación Conceptual	Propiedades en nivel M2	Propiedades en nivel M1
Anidamiento	E1 contiene E2	E1 \blacklozenge — E2	<u>E1</u> \blacklozenge — <u>E2</u>
Generalización / Herencia	E1 generaliza E2	E2 — \rightarrow E1	E2 — \triangleright E1
Importación	E1 importa E2	E1 — \rightarrow E2	ninguna
Agrupamiento	E1 agrupa E2	E1 — \rightarrow E2	<u>E1</u> \diamond — <u>E2</u> o ninguna

Tabla C-1. Mecanismos disponibles en MOF para la modularización y reutilización.

En la Tabla C-1 se muestra un resumen de las propiedades de estos cuatro mecanismos de composición de Paquetes MOF. La simbología utilizada es la de UML, de forma que un rombo relleno significa composición, un rombo hueco significa agregación, un triángulo hueco significa herencia, y una flecha discontinua significa dependencia. Notar que E1 y E2 representan cosas diferentes en cada columna: mientras que en las columnas 2 y 3 se refieren a Paquetes MOF en un metamodelo del nivel M2, en la columna 4 representan instancias de Paquetes (si aparecen subrayados) o sus tipos.

C.2.1.4. Tipos de Datos MOF.

El constructor MOF para especificar tipos de metadatos que no tengan identidad de objetos es Tipo de Dato (*DataType*). El sistema de tipos de datos de MOF está basado en el de CORBA (OMG, 2002c), incluyendo los tipos de datos nativos de CORBA con una colección de los tipos primitivos simples habituales y de tipos enumerados, estructurados y colecciones. Adicionalmente, también se pueden reutilizar tipos externos mediante un interfaz idéntico al equivalente de CORBA.

El uso más habitual de los tipos de datos en MOF es para asignar tipo a los atributos y los parámetros de las operaciones. Aunque el modelador puede elegir entre modelar metadatos como objetos (nodos) o como valores (mediante tipos de datos), en general, el uso de Tipos de Datos genera representaciones más sencillas pero que tienen la desventaja de que no pueden ser utilizados como Finales de Asociación, es decir, no se pueden crear enlaces a un valor.

C.2.1.5. Otros Constructores MOF.

Los constructores anteriores permiten definir información de metadatos que incluye nodos (Clases) con propiedades asociadas (Atributos y Tipos de Datos) y las interrelaciones entre los nodos (Asociaciones). El Modelo MOF provee varios constructores de modelado adicionales, que complementan a los cuatro anteriores:

- Restricciones (*constraints*): Este constructor permite que los metadatos estén limitados por un conjunto de fórmulas bien formadas que tienen sentido semántico. Las Restricciones sirven para asociar reglas de consistencia a otros componentes de metamodelado. Tienen las siguientes propiedades: nombre, lenguaje (cuál es el lenguaje utilizado para expresar la regla de consistencia), expresión de restricción (en el lenguaje anterior), política de evaluación (inmediata o diferida), y conjunto de elementos restringidos (Clases, Asociaciones, Tipo de Datos, Parámetros o Paquetes entero). Aunque se pueden utilizar diversos lenguajes para expresar restricciones, es conveniente representarlas mediante el lenguaje formal OCL (*Object Constraint Language*), que forma parte de la propia norma UML (capítulo 6 de la versión 1.4). Con este lenguaje es posible especificar invariantes, precondiciones, postcondiciones o, por ejemplo, indicar cuando una referencia a un objeto puede ser nula.
- Constantes (*constants*): Una Constante define una correspondencia entre un nombre y un valor constante.
- Excepciones (*exceptions*): Este constructor permite declarar la invocación (*signature*) de una excepción, que puede ser activada por una Operación.
- Etiquetas (*tags*): Las Etiquetas son el mecanismo para extender o modificar metamodelos MOF “puros”, es decir, sirven para particularizar y extender el lenguaje de metamodelado. Una Etiqueta contiene lo siguiente: un nombre, un identificador del tipo de etiqueta, una colección de cero o más valores asociados, y un conjunto de elementos a las que la etiqueta está vinculada. Asociar una Etiqueta a un elemento de metamodelado modifica el significado de dicho tipo de forma parecida a como lo hacen los estereotipos cambian el semántica de una clase UML.

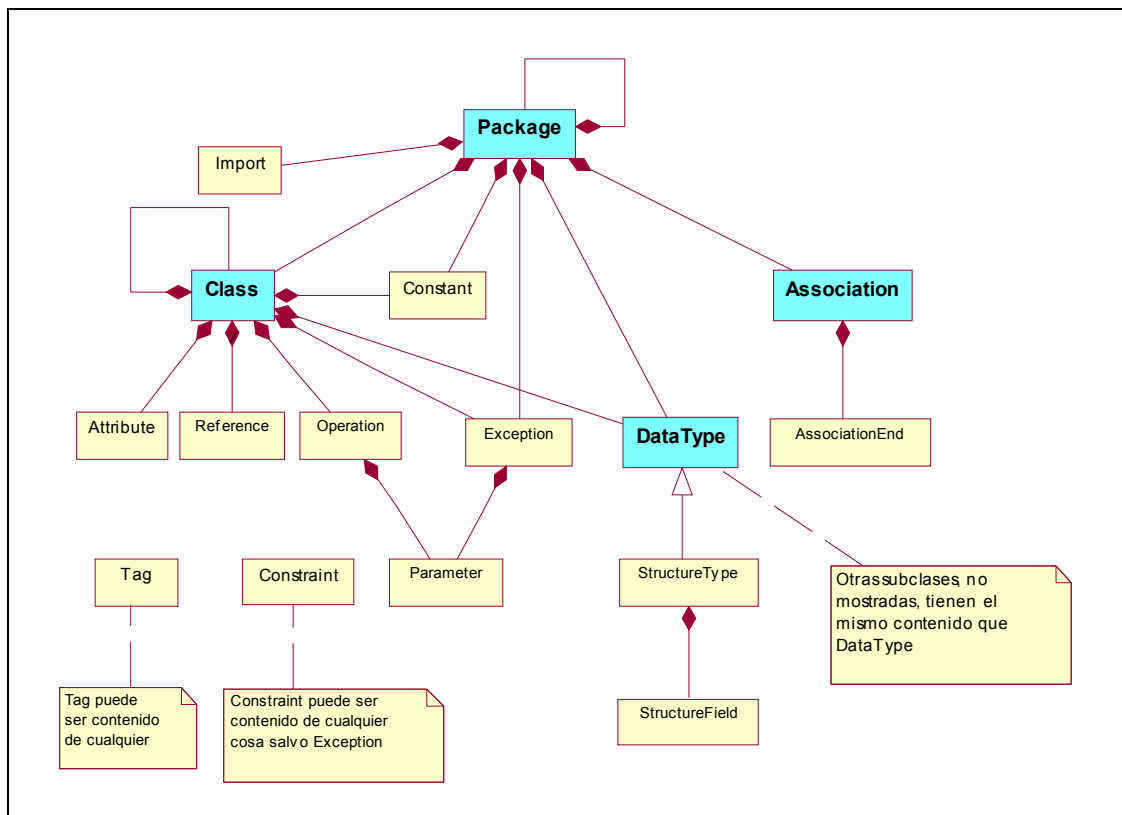


Figura C-3. Principales relaciones entre las clases del modelo MOF.

C.2.2. Relaciones entre los Diversos Constructores.

Las mayoría de las diversas relaciones que pueden existir entre los constructores de metamodelado incluidos en el Modelo MOF ya han sido citadas a lo largo de los apartados anteriores. En la Figura C-3 se muestran todas ellas de forma resumida. Los cuatro constructores principales han sido resaltados.

En dicha figura se ve de forma clara que un Paquete contiene otros Paquetes (por anidamiento o importación), Clases, Constantes, Excepciones, Tipos de Datos y Asociaciones. Igualmente, una Clase contiene otras Clases, Atributos, Referencias, Operaciones, Excepciones, Tipos de Datos y Constantes; las Operaciones y Excepciones contienen Parámetros; y Las Asociaciones contienen Finales de Asociación. Además, los Tipos de Datos pueden ser, entre otros, Estructurados, en cuyo caso, están formados por Campos Estructurados. Por último, las Etiquetas y restricciones pueden estar contenidas en cualquier elemento, salvo Excepciones en el caso de las segundas.

D. XMI: XML Metadata Interchange.

El principal propósito de la norma XMI (*XML Metadata Interchange*), propuesta por el OMG (2002a), es facilitar el intercambio de metadatos, en entornos distribuidos heterogéneos, entre diferentes tipos de herramientas software y, en especial, entre herramientas de modelado basadas en UML y repositorios de metadatos basados en la propuesta MOF (ver anexo C). Para ello, XMI integra tres normas que son claves en la industria actual de sistemas de información. Estos estándares de facto son:

- XML:** Lenguaje de Marcas Extensible (*eXtensible Markup Language*), estándar del *World Wide Web Consortium* que especifica un formato universal para documentos y datos estructurados en la web (W3C, 2000).
- UML:** Lenguaje Unificado de Modelado (*Unified Modeling Language*), estándar propuesto por el OMG (2001a).
- MOF:** Facilidad para Meta-Objetos (*Meta-Object Facility*), ya comentado.

La integración de estos estándares en XMI une lo mejor de las tecnologías sobre metadatos y modelado de OMG y W3C, permitiendo que los desarrolladores de sistemas de información puedan compartir modelos y otros metadatos vía Internet (Figura D-1). El uso combinado de las tres normas anteriores permite reducir de forma drástica la cantidad de formatos de intercambio de metadatos.

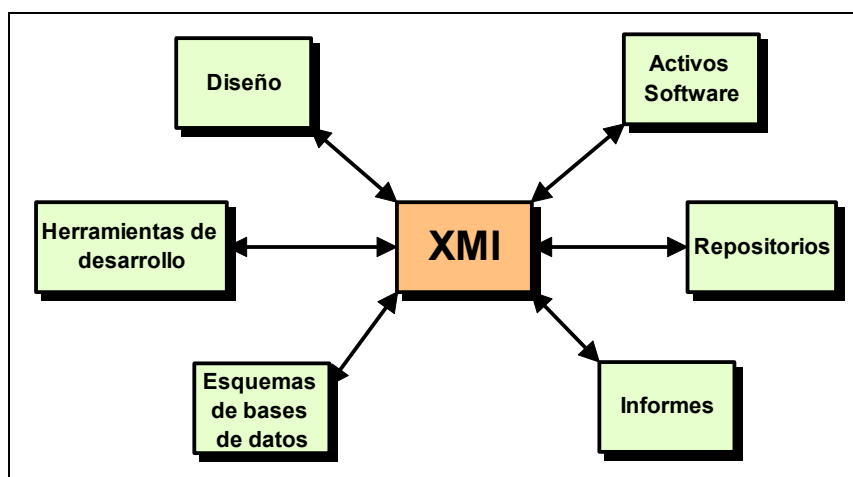


Figura D-1. Intercambio abierto de metadatos entre distintos tipos de herramientas mediante XMI.

XMI permite que los metadatos puedan ser intercambiados como flujos o archivos con un formato estándar basado en XML. La arquitectura completa ofrece un amplio rango de posibilidades de implementación a los desarrolladores de herramientas, repositorios y marcos de trabajo orientados a objetos. Para ello, la especificación XMI define los siguientes aspectos:

- Un conjunto de reglas de producción de DTD's para transformar metamodelos basados en MOF en DTD's (*Document Type Definition*) en formato XML. Estos DTD's se utilizan como sintaxis para la construcción de documentos XMI. Por tanto, dado un

metamodelo MOF (nivel M2 de la arquitectura presentada en el anexo C) expresado en XMI, el DTD asociado permite validar modelos MOF (nivel M1).

- Un conjunto de reglas de producción de documentos XMI, que sirven para la codificación y decodificación de metadatos basados en MOF, es decir, para representar los metamodelos y modelos MOF (niveles M2 y M1) en formato XMI.
- Una serie de principios de diseño para los DTD's y flujos XML basados en XMI.
- Archivos con los DTD's específicos del metamodelo de UML y del Modelo MOF.

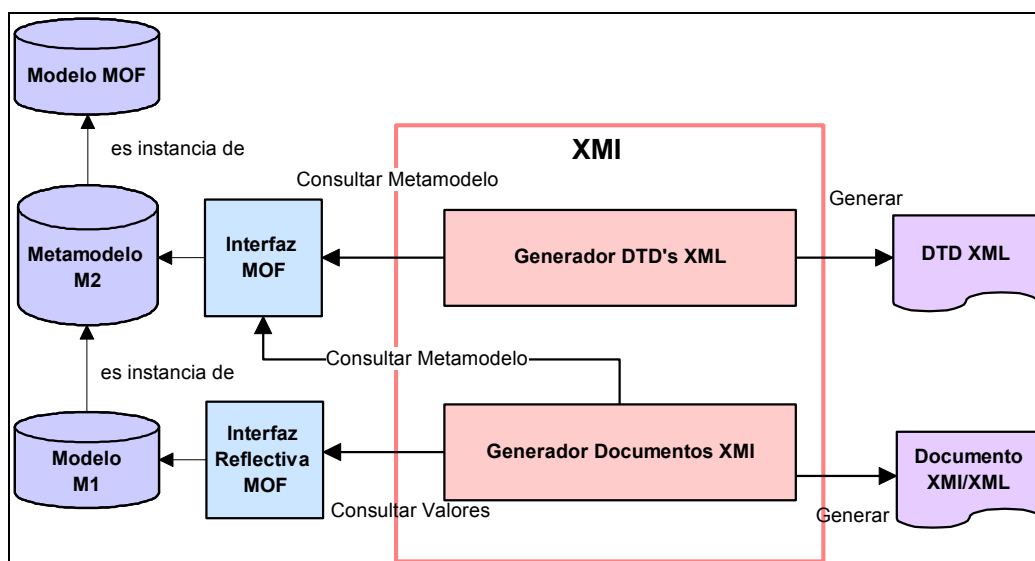


Figura D-2. Relación entre MOF y XMI.

D.1. Relación entre XMI y MOF.

Como ya se ha comentado, XMI es un formato de intercambio de metadatos definido en base a MOF. De hecho, la parte básica de XMI consiste en un par de correspondencias, por un lado, entre metamodelos del nivel M2 de MOF y DTD's XML y, por otro, entre metadatos MOF (modelos del nivel M1) y documentos XML. En la Figura D-2 se muestra un esquema representando esta relación entre MOF y XMI. Como se puede apreciar, los dos módulos que conforman la especificación XMI (generador de DTD's XML y generador de documentos XMI) acceden a las interfaces proporcionadas por MOF (llamadas "Interfaz MOF" e "Interfaz Reflectiva MOF" respectivamente) para obtener la información sobre los modelos y metamodelos y, a partir de ellos, construir los documentos XML (que representan los modelos MOF del nivel M1) y los DTD's asociados (que describen los metamodelos MOF del nivel M2). Por tanto, todo documento XMI permite establecer una correspondencia entre dos niveles de abstracción consecutivos (niveles M3-M2 o niveles M2-M1 de la arquitectura conceptual de MOF).

Por ejemplo, en la Figura D-3 se observa un fragmento de código XMI que representa la correspondencia entre un metamodelo de nivel M2 (en este caso el de UML) y un modelo del nivel M1 (el modelo UML de una empresa cualquiera). Como se puede observar, los elementos del metamodelo M2 ('class', 'attribute', 'multiplicity', etc.) se representan como etiquetas XML (encerrados entre "< >") y los elementos del modelo M1 ('Negocio', 'Cliente', 'CIF', etc.) se representan como datos XML.

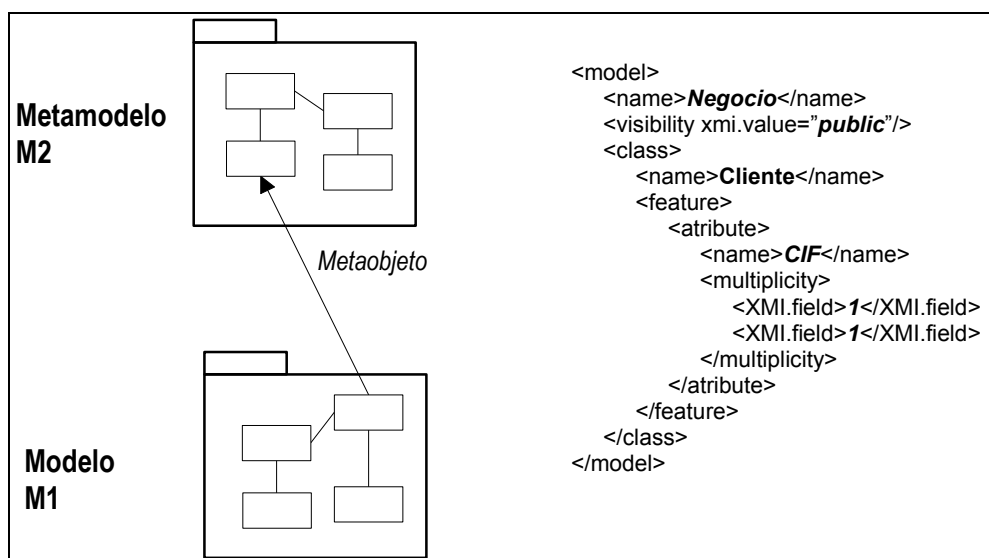


Figura D-3. Ejemplo de código XMI: una correspondencia M2-M1.

Cualquier repositorio o herramienta que pueda codificar y decodificar flujos XMI puede intercambiar metadatos con otros repositorios o herramientas que tengan la misma capacidad. Además, XMI proporciona una posible forma de intercambio de metadatos entre repositorios cuyos metamodelos no están basados en MOF. Este intercambio puede ser realizado mediante correspondencias 'ad hoc' (basadas en la experiencia del modelador) entre el documento XMI y el metamodelo nativo del repositorio, es decir, tratando de hacer corresponder adecuadamente a cada elemento del meta-metamodelo del repositorio no basado en MOF un elemento del modelo MOF.

D.2. Diseño de los DTD's XMI.

Un DTD XML proporciona el medio necesario para que un procesador XML pueda validar la sintaxis e incluso parte de la semántica de un documento XML (W3C, 2000). Aunque el uso de DTD's es opcional, ya que en un documento XML no es obligatoria una referencia a un DTD, XMI recomienda su uso porque permiten realizar la validación automática de los documentos XMI/XML que representan metamodelos. Por esta causa, la especificación XMI proporciona reglas para generar DTD's de metamodelos MOF de nivel M2 en correspondencia con el Modelo MOF del nivel M3 (que es el meta-metamodelo).

A pesar de que la validación XMI no realiza una verificación completa desde el punto de vista de la adecuación de los datos al metamodelo, debido a que un DTD no puede representar toda la semántica de un metamodelo, el uso de DTD's reduce la carga de trabajo de programación para realizar dicha validación. La validación XML puede determinar si los elementos XML requeridos por la especificación XMI están presentes en el documento XML que representa dicho metamodelo (y sus datos). Un ejemplo de validación podría ser comprobar que los atributos requeridos en los elementos XML tienen valores del tipo adecuado o si los valores son correctos.

En general, los requerimientos del DTD asociado con un documento XMI son los siguientes:

- Todos los elementos XML definidos por la especificación XMI deben estar declarados en el DTD.
- Cada elemento constructor de un metamodelo (clase, atributo, asociación, etc.) debe tener su correspondiente declaración XML en forma de entidad, atributo, etc.
- Cualquier elemento XML que representa extensiones al metamodelo debe ser declarado en el DTD como una entidad interna o externa.

D.2.1. Estructura General de un DTD en XMI.

Cada metamodelo M2 es representado en XMI mediante un DTD. Estos DTD's XMI contienen las siguientes partes:

- Una instrucción indicando la versión del documento XML de la forma '`<? XML versión="1.0" ?>`'.
- Una declaración opcional sobre la codificación del documento. Por ejemplo: '`<? XML version = "1.0" ENCODING="UCS-2" ?>`'.
- Una declaración opcional indicando un DTD externo o declaración de un DTD interno, por ejemplo, '`<! DOCTYPE XMI SYSTEM "http://www.xmi.org/xmi.dtd">`'.
- Declaraciones específicas de un metamodelo.

Las declaraciones específicas del metamodelo tienen al elemento `<XMI>` como el de más alto nivel. A partir de este elemento se estructura el resto de elementos. A continuación se describen los elementos que conforman un documento XMI de forma más detallada. La declaración para el elemento `<XMI>` en el DTD sería:

```
<!ELEMENT XMI (XMI.header?,
XMI.content?,
XMI.differences*,
XMI.extensions*) >

<!ATTLIST XMI
xmi.version CDATA #FIXED "1.1"
timestamp CDATA #IMPLIED
verified (true | false) #IMPLIED
>
```

En esta declaración aparecen tres atributos: 'xmi.version' tiene el valor de la versión actual de la especificación XMI; 'timestamp' indica la fecha y hora en que los metadatos fueron escritos; y 'verified' indica si los metadatos ya han sido verificados. Los elementos que se estructuran a partir del elemento raíz `<XMI>` son los siguientes: encabezamiento (*header*), contenido (*content*), diferencia (*difference*) y extensiones (*extensions*). A continuación se describe su estructura.

D.2.1.1. Encabezamiento: XMI.header.

El elemento ‘XMI.header’ contiene los elementos XML que identifican el modelo, metamodelo y meta-metamodelo, así como un elemento opcional con información adicional sobre los metadatos (como autor, etc..). Su declaración es:

```
<!ELEMENT XMI.header (XMI.documentation?,
    XMI.model*,
    XMI.metamodel*,
    XMI.metametamodel* ,
    XMI.import*) >
```

A su vez, el elemento ‘XMI.documentation’ contiene información sobre los metadatos: el propietario, la persona de contacto, descripciones largas y cortas, la herramienta exportadora que creó los metadatos, la versión de la herramienta, y el copyright u otras noticias legales correspondientes. También es posible incluir otro tipo de información en formato texto. Su declaración es:

```
<!ELEMENT XMI.documentation (#PCDATA | XMI.owner | XMI.contact |
    XMI.longDescription | XMI.shortDescription | XMI.exporter |
    XMI.exporterVersion | XMI.notice)* >
<!ELEMENT XMI.owner ANY >
<!ELEMENT XMI.contact ANY >
<!ELEMENT XMI.longDescription ANY >
<!ELEMENT XMI.shortDescription ANY >
<!ELEMENT XMI.exporter ANY >
<!ELEMENT XMI.exporterVersion ANY >
<!ELEMENT XMI.exporterID ANY >
<!ELEMENT XMI.notice ANY >
```

‘XMI.model’ identifica el modelo del nivel M1. Es posible que los datos representen varios modelos diferentes, en cuyo caso, habrá varios elementos de este tipo. Su declaración es:

```
<!ELEMENT XMI.model ANY>
<!ATTLIST XMI.model %XMI.link.att;
    xmi.name CDATA #REQUIRED xmi.version CDATA #REQUIRED >
```

donde los atributos ‘XMI.name’ y ‘XMI.version’ son el nombre y la versión del modelo que es descrito en detalle en el elemento XMI.content.

De forma parecida, el elemento ‘XMI.metamodel’ identifica el metamodelo del nivel M2 al que corresponden los datos del modelo anterior. Es posible que los metadatos a ser transmitidos representen varios metamodelos de información. En este caso habrá varios elementos de este tipo. Su declaración es:

```
<!ELEMENT XMI.metamodel ANY>
<!ATTLIST XMI.metamodel %XMI.link.att;
    xmi.name CDATA #REQUIRED xmi.version CDATA #REQUIRED >
```

correspondiendo los atributos ‘XMI.name’ y ‘XMI.versión’ al nombre y versión del metamodelo respectivamente.

El elemento 'XMI.metametamodel' identifica el meta-metamodelo del nivel M3 al que corresponden los datos de un metamodelo. En el ámbito del trabajo de esta tesis, este meta-metamodelo será el Modelo MOF. Su declaración es:

```
<!ELEMENT XMI.metametamodel ANY>
<!ATTLIST XMI.metametamodel %XMI.link.att;
    xmi.name CDATA #REQUIRED xmi.version CDATA #REQUIRED >
```

refiriéndose 'XMI.name' y 'XMI.version' al nombre y versión del meta-metamodelo respectivamente.

Por último, 'XMI.import' identifica los documentos adicionales que son necesarios para procesar el documento actual. Su declaración es:

```
<!ELEMENT XMI.import ANY>
<!ATTLIST XMI.import %XMI.link.att;
    xmi.name CDATA #REQUIRED xmi.version CDATA #REQUIRED >x
```

D.2.1.2. Contenido: XMI.content.

El elemento 'XMI.content' contiene los metadatos, que pueden representar información correspondiente a un modelo (nivel M1) o a un metamodelo (nivel M2). Su declaración es:

```
<!ELEMENT XMI.content ANY >
```

A continuación presentamos las reglas para representar los principales elementos de metamodelado de MOF (ver anexo C para más detalle).

D.2.1.2.1. Especificación de Clases.

Cada clase del metamodelo se representa en el DTD por un elemento XML cuyo nombre es el nombre de la clase. La definición de este elemento lista los atributos de la clase, las referencias a finales de asociación relacionados con la clase y las clases contenidas por esta clase (explícitamente o mediante asociaciones de composición).

A su vez, cada atributo de una clase se representa en el DTD por un elemento XML cuyo nombre es el nombre del atributo. Además, los atributos que tienen datos de tipos primitivos o enumerados se representan en el DTD como una declaración de atributo XML. Los atributos se incluyen en el elemento que representa la clase a la que pertenecen, sin importar el orden. Además, cada asociación (tanto las de agregación como las que no) entre clases del metamodelo se representan con dos elementos XML que, a su vez, representan los dos roles de los correspondientes finales de asociación (recordar que en MOF todas las asociaciones son binarias).

De esta manera, si se tiene una clase con nombre 'C' y que tiene atributos, asociaciones y relaciones de composición, su declaración en el DTD sería:

```
<!ENTITY % cProperties 'propertiesForC' >
<!ENTITY % cAssociations 'associationsForC'>
<!ENTITY % cCompositions '| XMI.extension | compositionsForC'>
<!ELEMENT C (%cProperties | %cAssociations | %cCompositions)* >
<!ATTLIST C %XMI.element.att; %XMI.link.att; >
```

D.2.1.2.2. Especificación de Atributos.

La representación de un atributo de una clase puede que use elementos XML o bien atributos XML. Si el atributo del metamodelo es de tipo primitivo o enumeración, los elementos XML declarados son atributos XML; en caso contrario, son declarados como elementos XML. A continuación se muestra un ejemplo de este último caso, considerando que el atributo tiene por nombre ‘A’ y que pertenece a la clase ‘C’:

```
<!ELEMENT C.A ( type specification) >
```

En cambio, cuando ‘A’ es un atributo con valores enumerados o primitivos se declararía de esta otra manera:

```
<!ATTLIST C.A XML.value (enum1 | enum2 | ...) #REQUIRED >
```

D.2.1.2.3. Especificación de Asociaciones.

Cada rol de asociación se representa como una entidad XML, un elemento XML y un atributo XML. La multiplicidad del rol no se usa para crear una declaración XMI (es decir, esta información semántica se pierde). De esta manera, la declaración de una asociación con un rol llamado ‘R’ para una clase ‘C’ del metamodelo sería:

```
<!ENTITY % cAssociations ‘R’ >
<!ELEMENT R ( content)* >X
```

Adicionalmente a lo anterior, cada final de asociación que representa una relación de composición se representa por una entidad XML y un elemento XML. El modelo de contenidos de dicho elemento XML es el del elemento correspondiente a la clase y los elementos XML correspondientes a cada una de las subclases de la clase. Por ejemplo, si una clase ‘C’ está compuesta por otras (y por tanto es la clase que está en el final de asociación que representa composición) y el otro final de asociación tiene un rol ‘R’ para una clase ‘C1’ con una subclase ‘C2’, la declaración en un DTD de XML sería:

```
<!ELEMENT r (C1 | C2)* >
<!ENTITY % cCompositions ‘XML.extension | R’ >X
```

D.2.1.3. Diferencias: XMI.differences.

Este elemento contiene los elementos XML que representan las diferencias con respecto a los datos base. Los usuarios pueden hacer uso de estas diferencias dentro del elemento ‘XMI.content’ o en una sección separada de diferencias. Los atributos de este elemento permiten hacer referencias a otros elementos mediante el uso de ‘Xlinks’, ‘Xpointers’, o ‘IDRefs’ (incluir referencia). Su declaración es:

```
<!ELEMENT XMI.differences (XMI.difference | XMI.add | XMI.delete | XMI.replace)* >
<!ATTLIST XMI.difference %XMI.element.att; %XMI.link.att; >
```

En esta declaración, los elementos incluidos significan lo siguiente:

- ‘XMI.add’ representa los elementos a añadir a los datos base. Su declaración es (el atributo ‘XMI.position’ indica la posición relativa de la adición respecto a los otros elementos):

```
<!ELEMENT XMI.add ANY >
<!ATTLIST XMI.add %XMI.element.att; %XMI.link.att; xmi.position CDATA "-1" >
```

- ‘XMI.delete’ representa los elementos a borrar respecto a lo datos base. Su declaración es:
`<ELEMENT XMI.delete EMPTY >`
`<!ATTLIST XMI.delete %XMI.element.att; %XMI.link.att; >`
- ‘XMI.replace’ representa los elementos a reemplazar en los datos base. Su declaración es (el atributo ‘XMI.position’ indica la posición relativa donde se tiene que efectuar la sustitución):
`<ELEMENT XMI.add ANY >`
`<!ATTLIST XMI.add %XMI.element.att; %XMI.link.att; xmi.position CDATA "-1" >`

D.2.1.4. Extensiones: XMI.extensions.

Cada DTD XMI dispone de un mecanismo para extender una clase de un metamodelo. Es posible incluir cualquier número de elementos de tipo ‘XMI.extension’ en el modelo de contenidos de cualquier clase. El elemento ‘XMI.extensions’ contiene los elementos XML que contienen metadatos que constituyen una extensión del metamodelo. Esta información podría incluir por ejemplo, información de presentación asociada con los metadatos. Su declaración es:

```
<ELEMENT XMI.extensions ANY >
<!ATTLIST XMI.extension xmi.extender CDATA #REQUIRED>
```

donde el elemento ‘XMI.extension’ contiene elementos XML, que a su vez contienen metadatos que son una extensión al metamodelo. Su declaración es:

```
<ELEMENT XMI.extension ANY >
<!ATTLIST XMI.extension %XMI.element.att; %XMI.link.att;
xmi.extender CDATA #REQUIRED xmi.extenderID CDATA #IMPLIED>
```

siendo ‘xmi.extender’ un atributo que indica el nombre de la herramienta que hace la extensión y ‘xmi.extenderID’ un identificador interno de dicha herramienta.

D.2.1.5. Referencias: XMI.reference.

El elemento ‘XMI.reference’ permite realizar referencias a otros elementos XML dentro de un atributo de tipo cadena o de un elemento de tipo ‘XMI.any’, que representa un tipo de datos no incluido en el metamodelo. Su declaración es:

```
<ELEMENT XMI.reference ANY >
<!ATTLIST XMI.reference %XMI.link.att; >
```

D.3. Generación de los Documentos XMI.

La especificación XMI define la forma de representar un modelo (del nivel M1) como un documento XML. Para ello especifica un conjunto de reglas de producción que se pueden aplicar sobre cualquier modelo cuyo metamodelo haya sido descrito usando MOF. Sin embargo, el modelo MOF no implica el uso de un constructor o mecanismo específico de definición dentro de un metamodelo de lo que será un modelo, es decir en el propio metamodelo M2 el metamodelador especifica cuales van a ser los constructores de los modelos, sin ningún tipo de restricción. Esto dota a los metamodeladores de gran flexibilidad a la hora de definir sus

modelos de información. XMI proporciona dos métodos diferentes de especificar los elementos de modelado que serán usados para generar un documento XMI:

- Composición de objetos (*Object Containment*); y
- Extensión de paquetes (*Package Extent*).

En el desarrollo de esta tesis se ha utilizado el primer método, razón por la cual se comenta más en detalle que el segundo.

D.3.1. Producción por Composición de Objetos.

La mayoría de los metamodelos se caracterizan porque están basados en una jerarquía de composición, de forma que los elementos de modelado de cualquier tipo se componen de otros elementos de modelado. En UML, por ejemplo, un modelo está compuesto por clases, casos de uso, paquetes, etc.; elementos que, a su vez, pueden tener otros componentes. Esta composición se define en los metamodelos usando Asociaciones MOF y especificando en uno de sus Finales de Asociación que es una agregación de tipo “compuesto” (*composite*). Esta composición debe cumplir la restricción de que un elemento no puede ser contenido en múltiples composiciones.

Las reglas detalladas para producir los documentos XMI son muy extensas para incluirlas en este documento. En cualquier caso, dichas reglas pueden ser consultadas en el documento original (OMG, 2002a), y por eso, a continuación, nos limitamos a realizar una presentación muy breve de ellas, que se completa con un pequeño ejemplo de su uso.

Dado un objeto que representa un elemento de alto nivel en una jerarquía de composiciones, XMI define las reglas para generar un documento XML que represente ese objeto y todos sus objetos contenidos según dicha jerarquía de composición. Estas reglas navegan a través de la jerarquía de composición para generar todo el documento. Es necesario conocer el metamodelo para generar las etiquetas XML que describen los metadatos (instancias del metamodelo) en el documento XMI.

Para cada metaobjeto (incluyendo el raíz) se genera un elemento XML que lo representa. La etiqueta de comienzo de dicho elemento incluye el nombre de la clase a la que pertenece la instancia del metaobjeto y un identificador único en el documento (XMI.id). Los nombres de los elementos XML están totalmente cualificados basándose en la descripción MOF del metamodelo, de tal manera que están formados por la secuencia de composiciones y contenidos (comenzando en el paquete más externo del metamodelo) separados por puntos.

Cada atributo de cada metaobjeto se representa como otro elemento XML con el nombre del atributo y su valor o valores. De forma similar se representan las referencias, asociaciones y demás elementos.

D.3.1.1. Ejemplo de Aplicación: Grafos.

Se trata de representar grafos simples. Para ello, se ha diseñado un metamodelo del nivel M2 que define un lenguaje (conjunto de constructores) para desarrollar grafos, es decir, modelos de grafos del nivel M1. En la Figura D-4 se muestra dicho metamodelo (una de las múltiples posibilidades existentes) representado mediante un diagrama de clases UML.

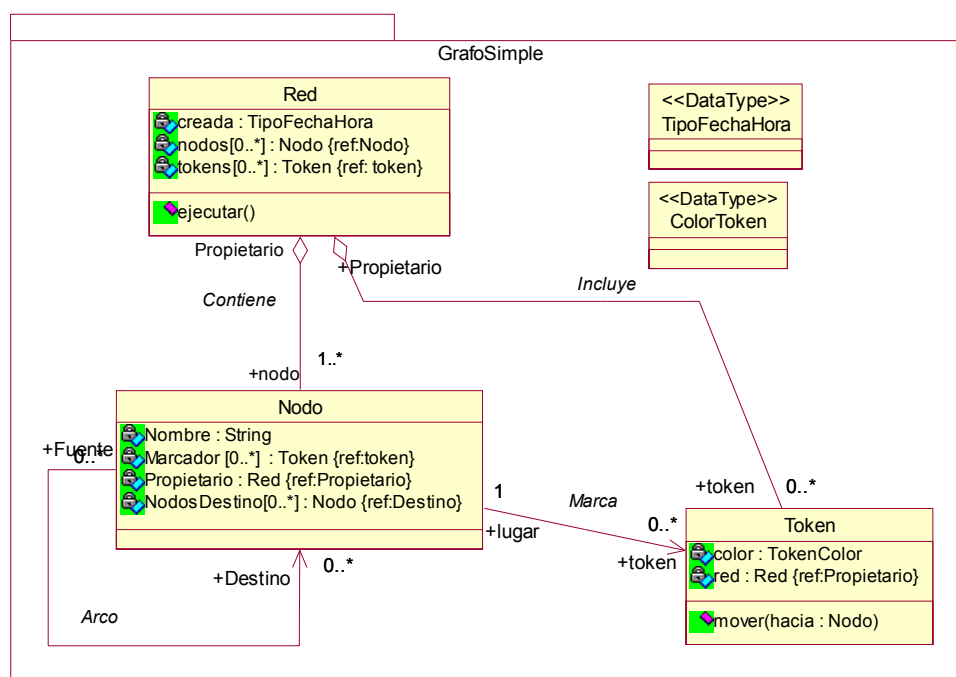


Figura D-4. Ejemplo de metamodelo para representación de grafos.

El primer paso para poder representar instancias del metamodelo, es decir, modelos de grafos, sería asociar cada elemento del metamodelo (nivel M2) con su correspondiente del Modelo MOF (nivel M3), por ejemplo, construyendo el DTD correspondiente según las reglas de diseño comentadas en el apartado D.2. A fines de ejemplo, basta con saber que existen las correspondencias entre los niveles M3 y M2 indicadas en la Tabla D-1.

Correspondencias M3-M2 en el metamodelo de grafos	
Elementos del Modelo MOF (M3)	Instancias (elementos del metamodelo M2)
Paquetes MOF	Grafo Simple
Clases MOF (atributos / referencias / operaciones)	Red (creada / nodos, tokens / ejecutar), Nodo (nombre / marcador, propietario, nodosdestino /), Token (color / red / mover)
Asociaciones MOF (finales de asociación)	Incluye (propietario, token), Contiene (propietario, nodo), Arco (fuente, destino), Marca (lugar, token)
Tipos de Datos MOF	ColorToken, TipoFechaHora

Tabla D-1. Correspondencias M3-M2 en el ejemplo de metamodelado de grafos.

Para completar el ejemplo, en la Figura D-5 se han representado una instancia concreta (modelo de grafo) del metamodelo anterior. En la parte izquierda se ha representado en forma de diagrama de clases UML y en la parte derecha utilizando la notación gráfica habitual para los grafos. En este último caso, instancias del metaobjeto Marca se han representado como círculos de color dentro del círculo mayor que representa cada nodo.

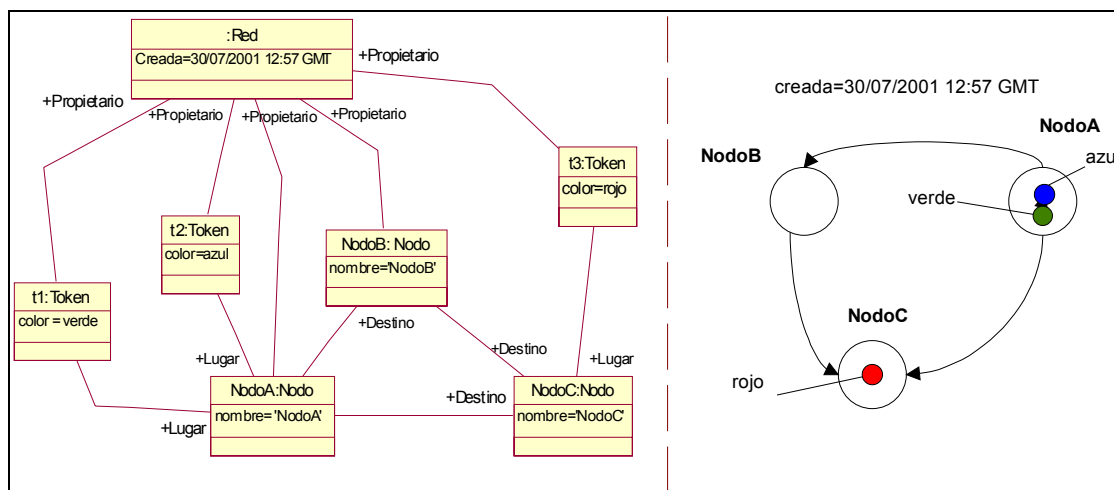


Figura D-5. Instancia del metamodelo y grafo que representa.

Un fragmento del documento XMI completo que representaría la instancia de la Figura D-5, según el método de composición de objetos, sería el siguiente:

```
<GrafoSimple.Red>

<GrafoSimple.Red XMI.id='a1'>

<GrafoSimple.Red.creado>
  <XMI.field>30/07/2001</XMI.field>
  <XMI.field>12:57 GMT</XMI.field>
</GrafoSimple.Red.creado>

.....

<GrafoSimple.Nodo XMI.id='a3'>
  <GrafoSimple.Nodo.Nombre>Nodo B</GrafoSimple.Nodo.Nombre>
  <GrafoSimple.Nodo.NodosDestino>
    <GrafoSimple.Nodo XMI.id='a4'>
      </GrafoSimple.Nodo.NodosDestino>
    </GrafoSimple.Nodo.NodosDestino>
  </GrafoSimple.Nodo>

.....

<GrafoSimple.Nodo XMI.id='a4'>
  <GrafoSimple.Nodo.Nombre>Nodo C</GrafoSimple.Nodo.Nombre>
  <GrafoSimple.Nodo.Marcador>
    <GrafoSimple.Nodo XMI.idref='a7'>
      </GrafoSimple.Nodo.Marcador>
    </GrafoSimple.Nodo.Marcador>
  </GrafoSimple.Nodo>

.....

</GrafoSimple.Red>
```

En este documento XMI, la instancia de la clase Red está representada por el texto ‘<GrafoSimple.Red XMI.id='a1'>’. Igualmente, el atributo “creado” de dicha instancia de Red está representado mediante un tipo estructurado. En esta situación, cada campo se representa como un elemento XML aparte con la marca <XMI.field> encerrando los valores de cada campo entre dichas marcas.

D.3.2. Producción por Extensión de Paquetes.

No siempre es posible o útil representar un conjunto de elementos de modelado mediante jerarquías de composición como las comentadas en el método anterior. Por esta razón, XMI define un segundo método para generar los documentos XMI a partir de los elementos de modelado, que está basado en extender los Paquetes, que son los elementos de más alto nivel englobando a los demás. Con este propósito, MOF proporciona un conjunto de interfaces que representan las características específicas de los paquetes, las clases y las asociaciones. Estas interfaces se agrupan en un módulo denominado Reflexivo (*Reflective Module*), que forma parte de la propia norma MOF (OMG,2002b). Como indica su nombre, este módulo aprovecha las capacidades de reflexión de los elementos del Modelo MOF para conseguir la información necesaria de un metamodelo o modelo.

En casos sencillos, este método produce los mismos resultados que el de composición de objetos, pero esto deja de cumplirse para modelos complejos. En general, la producción por composición de objetos es más apropiada que la producción por extensión de paquetes, salvo cuando se produce alguna de las siguientes circunstancias:

- más de una jerarquía de agregación necesita ser intercambiada; o
- hay interconexiones entre jerarquías de agregación separadas que necesitan ser replicadas; o
- existen atributos cuyo valor está definido a nivel de clase (en vez de instancia) que necesitan ser replicados.

E. Formalismo REFSENO para Representación de Ontologías.

REFSENO (*Representation Formalism for Software Engineering Ontologies*) es, como su nombre indica, un formalismo diseñado para la representación de ontologías de ingeniería del software mediante tablas, texto y, opcionalmente, diagramas; por lo que las ontologías obtenidas son semi-formales. REFSENO ha sido desarrollado en el Fraunhofer-IESE por Tautz y Von Wangenheim (1998).

Mediante REFSENO, el conocimiento de ingeniería del software puede ser representado por una ontología (conocimiento del dominio) y por instancias de los conceptos de dicha ontología (conocimiento de contexto específico). Para ello, se utilizan un conjunto de primitivas epistémicas que integran ideas de varios campos diferentes: bases de datos, razonamiento basado en casos y bases de conocimiento. Además, el formalismo es orientado a objetos. Es importante tener en cuenta que las ontologías definidas con REFSENO sirven para gestionar el conocimiento y no para implementar asistentes inteligentes.

REFSENO también propone un método de diseño de ontologías que está basado en una adaptación de la metodología “*Methontology*” (Fernández et al, 1997).

E.1. Primitivas Epistémicas.

A continuación se definen las diversas primitivas epistémicas utilizadas en el formalismo, haciendo especial hincapié en las utilizadas en el desarrollo de esta tesis, es decir, en aquellas que no tienen utilidad únicamente para una hipotética implementación futura o para fines de consultas a una base de conocimiento que pudiera almacenar la ontología. Para cada primitiva se indican los siguientes aspectos: nombre, ejemplo, sinónimos (si son conocidos), definición, descripción (explicación de los componentes incluidos en la definición), y representación (modo de representar este tipo de primitivas, es decir, sintaxis utilizada)²⁹.

E.1.1. Conceptos.

Un concepto modela entidades de ingeniería del software (un plan de aseguramiento de calidad, un modelo de proceso, un documento de diseño, un módulo de código, etc.) o es necesario para propósitos de modelado. Es sinónimo de tipo de artefacto, clase (en UML), entidad o marco (*frame*). Está definido por una 9-tupla con las siguientes propiedades:

- *Nombre*: Es único en la ontología.
- *Extensión*: Conjunto de todas las instancias pertenecientes al concepto.

²⁹ REFSENO también incluye diagramas como modos de representación alternativos, pero no son comentados en este documento porque en la realización de esta tesis se han sustituido por diagramas de clases UML, tal como se señala en el capítulo 5.

- *Intensión*: Conjunto de todos los atributos (de dos categorías, terminales y no terminales) que caracterizan a todas las instancias de la extensión. Los atributos pertenecen a una de las tres capas siguientes: artefacto, interfaz o contexto (ver apartado E.1.2.1).
- *Sim-artef*, *Sim-i/f*, *Sim-ctxt*: Son tres funciones de similaridad entre instancias (una por cada capa de atributos en la intención) utilizadas con fines de consulta.
- *Aserción*: Una condición que todas las instancias de la extensión deben cumplir.
- *Precondición*: Una condición que se debe cumplir antes de insertar o cambiar instancias.
- *Descripción*: Un texto que define la entidad.
- *Propósito*: Justificación de la existencia del concepto en la ontología, es decir, las tareas que necesitan el concepto. Para los conceptos que representan a entidades de ingeniería del software suele consistir en un escenario o caso de uso.
- *Usuarios*: Describe los roles que usarán el concepto.

Los conceptos, su intensiones y sus extensiones se representan en tres tipos de tablas diferentes llamadas, respectivamente, glosario de conceptos, tabla de atributos (apartado E.1.2) y tabla de instancias (apartado E.1.4).

El glosario de conceptos (ver ejemplo en la Tabla E-2) es una lista alfabética de todos los conceptos de la ontología. Para cada concepto, de las diez propiedades que definen su tupla, se incluyen sólo su nombre, descripción, propósito y usuarios.

E.1.2. Atributos.

Existen dos categorías de atributos: terminales, que equivalen a los atributos de las entidades en el modelo entidad-interrelación o de las clases en UML, y no terminales, que equivalen a las interrelaciones del modelo entidad-interrelación y a las asociaciones en UML.

E.1.2.1. Atributos Terminales.

Los atributos terminales modelan las propiedades de las entidades. Sus sinónimos son dimensión, elemento de datos, característica o propiedad. Están definidos por la 9-tupla siguiente:

- *Nombre*: Es único para la intención de un concepto.
- *Descripción*: Un texto con el significado del atributo.
- *Cardinalidad*: Un rango que indica el número mínimo y máximo de valores que el atributo puede tener. Se utiliza la misma notación que en UML.
- *Tipo*: Indica el tipo del atributo.
- *Valor por defecto*: Utilizado cuando se insertan nuevas instancias sin haber asignado valor al atributo.
- *Obligatoriedad*: Indica si es obligatorio que el atributo tenga valor para poder insertar una instancia nueva.
- *Valor inferido*: Para los atributos derivados, indica la fórmula para calcular su valor.

- *Atributos inferidos*: Lista de los atributos cuyo valor se calcula a partir del valor de este atributo. Cada atributo se indica por un par “[concepto]:[atributo]”.
- *Peso estándar*: Utilizado por las funciones de similaridad con fines de consulta.

Los atributos terminales se representan en una tabla de atributos para cada concepto, que contiene una fila por cada atributo, indicando su capa (artefacto, interfaz o contexto) y todas las propiedades de la 9-tupla anterior (ver ejemplos en Tabla E-3, Tabla E-4 y Tabla E-5). Los atributos se ordenan alfabéticamente dentro de cada capa. Además, la tabla incluye un encabezamiento con los nombres del concepto y del super-concepto. Los atributos del super-concepto son heredados por el concepto, es decir, la intensión de un concepto incluye a todos los atributos de su tabla más los incluidos en la intensión de su super-concepto. Existe un super-concepto raíz llamado “CONCEPT” para referirse a los conceptos que realmente no heredan de otro concepto. La tabla de atributos también incluye un pie con las funciones de similaridad, la precondition y la aserción.

Cada atributo de un concepto pertenece a una de las siguientes capas (Basili y Rombach, 1991):

- a) Capa de *artefacto* (artef): formada por los atributos que caracterizan las instancias propiamente dichas; por ejemplo, el autor y el lenguaje de programación de un módulo de código.
- b) Capa de *interfaz* (i/f): formada por los atributos que caracterizan cómo una instancia puede ser integrada en el sistema circundante. Ejemplos de este tipo son los parámetros y las variables globales de un módulo de código.
- c) Capa de contexto (ctxt): cuyos atributos caracterizan el entorno en el que las instancias son aplicadas; por ejemplo, la aplicación a la que pertenece un módulo de código.

E.1.2.2. Atributos no Terminales.

Junto con los atributos terminales, los atributos no terminales también son parte de la intensión de un concepto. Esta categoría de atributos modela la forma en que una entidad de ingeniería del software se relaciona con otras entidades, por ejemplo, para representar cómo un “objetivo de calidad GQM” está relacionado con un “plan de calidad”. Otros sinónimos son los punteros, los enlaces o las referencias. Un atributo no terminal está definido por una 11-tupla, cuyas propiedades son:

- *Nombre*: Es único para la intensión de un concepto.
- *Clase*: Indica la clase de asociación, por ejemplo, “es-un” o “es-parte-de”.
- *Destino*: Es el concepto asociado, de forma que los valores de un atributo no terminal son un conjunto de instancias de dicho concepto de destino.
- *Atributo inverso*: El atributo correspondiente del concepto de destino.
- *Descripción, Cardinalidad, Valor por defecto, Obligatoriedad, Valor inferido, Atributos inferidos y Peso estándar*: Se definen igual que para los atributos terminales. Utilizado por las funciones de similaridad con fines de consulta.

Los atributos no terminales se representan en la misma tabla de atributos que los terminales, pero con la salvedad de que en la columna “tipo” se indican la clase, el concepto de destino y el atributo inverso, mediante un texto de la forma “<clase>[destino].[atributo inverso]”. Además, el valor por defecto se indica mediante un nombre de instancia del concepto de destino. En la Tabla E-3 se muestra un ejemplo de representación de atributos de esta categoría. Para mejorar su visibilidad, REFSENO recomienda representar estos atributos también mediante un diagrama de clases similar a los de UML.

Las clases de atributos no terminales son análogas a los tipos de los atributos terminales, y permiten definir la semántica de las interrelaciones entre entidades, por ejemplo, si la interrelación entre un plan de calidad y una medición es de la clase “incluye-a”, significará que un plan de calidad incluye mediciones. Estas clases están definidas por una 5-tupla con las siguientes propiedades:

- *Nombre*: Es único en la ontología.
- *Nombre inverso*: Las interrelaciones modeladas por los atributos no terminales pueden ser leídas en dos direcciones diferentes. Este nombre se refiere a la segunda. Por ejemplo, el nombre podría ser “es-componente-de” y el nombre inverso sería “está-compuesto-por”.
- *Propósito*: Justificación de la existencia de esta clase de atributo no terminal en la ontología. Suele consistir en un conjunto de escenarios de uso que muestran su utilidad.
- *Estructura*: Las instancias de los conceptos se relacionan por medio de las clases de atributos no terminales. Si consideramos que las instancias hacen el papel de nodos y los valores de atributos no terminales son flechas apuntando desde una instancia origen a otra destino, esta propiedad indica la clase de estructura que se pueden crear: un conjuntos de árboles (*trees*), un grafo acíclico dirigido (GAD), o un grafo (sin restricciones).
- *Propiedades*: Relaciona otras propiedades que pueda tener la estructura creada, por ejemplo, simetría, transitividad, etc.

En REFSENO existen 4 clases de atributos no terminales predefinidas, que son muy similares a las clases de asociaciones en UML:

- “*es-un*”: Su nombre inverso es “está-especializado-en”. Denota una especialización de un concepto. Su estructura es un árbol. Sus principales propiedades son la transitividad, la herencia simple, y que el nodo raíz del árbol es “CONCEPT”.
- “*es-instancia-de*”: Su inverso es “está-instanciado-en”. Es un caso especial de clase “*es-un*” que vincula una instancia con su concepto (en vez de dos conceptos). Su estructura es un árbol de sólo dos niveles, es decir, sin nodos intermedios.
- “*incluye-a*”: Que tiene como inverso a “es-parte-de”. Se refiere a una descomposición en la cual las partes pueden ser compartidas por más de un concepto. Su estructura es un GAD y cumple la propiedad de transitividad.
- “*está-compuesto-por*”: Cuyo nombre inverso es “es-componente-de”. Denota una agregación, es decir, una descomposición en la que los componentes sólo existen si existe el compuesto agregado.

Las clases de atributos no terminales adicionales a las anteriores se representan en una tabla de clases de atributos no terminales, que incluye una fila por cada clase con los valores de las cinco propiedades de su tupla (ver Tabla E-7).

E.1.3. Tipos.

Todos los atributos terminales son tipados. Los tipos de atributos modelan cualidades de las entidades de ingeniería del software (por ejemplo, número de líneas de código de un módulo) o se utilizan para especificar otras entidades (por ejemplo, el tipo “lenguaje de programación” de una entidad “módulo de código”). Un tipo está definido por la 5-tupla siguiente:

- *Nombre*: Es único para una ontología.
- *Supertipo*: Es el tipo del que un tipo hereda propiedades. Un tipo se puede diferenciar de su supertipo tan sólo en el rango de valores, en la unidad de medida o en la función de similaridad. El supertipo puede ser un tipo predefinido, otro tipo definido en la ontología o el tipo raíz denotado por “TYPE”.
- *Rango de valores*: Posibles valores de los atributos de este tipo. Existen tres valores predefinidos especiales: “indefinido”, “desconocido” y “n/a” (no aplicable).
- *Unidad de Medida*: Sólo es aplicable para tipos numéricos.
- *Sim*: función de similaridad definida con fines de consulta.

Existe una colección de tipos predefinidos a partir de los cuales se pueden derivar subtipos. Los tipos ordenados, es decir, con un orden predefinido entre sus valores, son “entero”, “real”, “texto”, “identificador”, “fecha”, “tiempo”, “marca de tiempo” y “símbolo ordenado”. Los tipos no ordenados son “booleano”, “símbolo”, “taxonomía de símbolos” e “intervalo”.

Los tipos se representan en una tabla de tipos, que incluye una fila por cada tipo (ver Tabla E-6). Los tipos están ordenados alfabéticamente por su nombre. Para cada tipo se indican los cinco valores de su tupla de definición. El rango de valores se puede representar como una lista de rangos no solapados, siendo cada rango una pareja de valores (mínimo, máximo), donde el * representa un $-\infty$ para el mínimo y $+\infty$ para el máximo. También se puede realizar una enumeración de los posibles valores. Para tipos numéricos los valores se escriben ordenados. Para tipos simbólicos, los valores se escriben entrecomillados y en una estructura que depende del supertipo del que se hereda:

- si el supertipo es “símbolo”, los símbolos se colocan alfabéticamente;
- si el supertipo es “símbolo ordenado”, los símbolos se ordenan de menor a mayor; y
- si el supertipo es “taxonomía de símbolos”, entonces los símbolos se colocan en una jerarquía textual formateada mediante sangrados, similar al índice de un libro.

Para los tipos simbólicos, es decir, aquellos cuyos valores son símbolos que tienen semántica asociada, la representación incluye un glosario de símbolos (ver Tabla E-8). El glosario de símbolos es una tabla, con las columnas tipo, símbolo y descripción, que muestra todos los símbolos de una ontología agrupados por el tipo (de atributo).

E.1.4. Instancias.

Afortunadamente, no es necesario que todas las instancias que forman la extensión de cada concepto se incluyan en una ontología. De hecho, lo normal es que sólo unas pocas (o ninguna) se tengan que considerar porque aportan un conocimiento de contexto específico interesante. Por ejemplo, se podrían incluir instancias de “medición” que son obligatorias y muy importantes para el éxito de un “programa de medición”. Sus sinónimos son caso, caracterización y objeto. Una instancia está definida por una 3-tupla con las propiedades siguientes:

- *Nombre*: Es único en la ontología.
- *Concepto*: Indica el concepto a cuya extensión pertenece la instancia. La intensión de la instancia coincide con la intensión de este concepto.
- *Valores*: La lista de valores de los atributos de la instancia.

Una instancia se representa por una tabla de instancia (ver Tabla E-1). Cada tabla de instancia permite conocer los valores de cada uno de los atributos de una instancia mediante las columnas *capa* (ver apartado E.1.2.1), *atributo* y *valor*. Además, en el encabezamiento de la tabla se indican el concepto y el nombre de la instancia.

Concepto: Medición GQM		
Instancia: Contador-Fallos-1		
Capa	Atributo	Valor
artef	Id	Contador-Fallos-1
	Definición	Cuenta el número de informes de fallo recibidos antes de la entrega
	Escala	Entero positivo
	Unidades	Número de ocurrencias
i/f	Asunción	Se supone que se recibe un informe de fallo por cada fallo detectado
	Procedimiento recogida datos	“desconocido”
ctxt	Plan GQM	“desconocido”

Tabla E-1. REFSENO: Ejemplo de una tabla de instancia.

E.1.5. Otras Primitivas.

Las fórmulas se utilizan, con fines de consulta, en funciones de similaridad, aserciones, precondiciones y valores inferidos. Permiten modelar, entre otros, los aspectos siguientes:

- similaridad entre artefactos de ingeniería del software,
- similaridad entre valores, y
- dependencias entre valores.

Una fórmula es similar a una regla. Consiste en un término matemático en el que las variables son atributos de conceptos. REFSENO incluye una colección de operadores, constantes, funciones y valores especiales (por ejemplo, “n/a”) predefinidos para poder utilizarlos en las fórmulas.

E.2. Caso de Uso: Ontología de los Productos.

Presentamos a continuación la aplicación del formalismo a la sub-ontología de los productos presentada en el capítulo 5. Las tablas se muestran completas. En primer lugar, mostramos el glosario de conceptos y las tablas de atributos.

Nombre	Descripción	Propósito	Usuarios
Artefacto	Parte de un producto software que es consumida, modificada o producida. Ejemplos: documento de especificación de requisitos, plan de calidad, módulo de clase, rutina, informe de pruebas, manual de usuario. Sinónimos: elemento software, producto de trabajo.	Definir la estructura interna y composición del software.	ingenieros, gestores
Producto	Software que está siendo mantenido. Sinónimo: Software.	Mantenerlo.	ingenieros, gestores, usuarios
Versión	Cada una de las versiones, <i>upgrades</i> , <i>releases</i> o actualizaciones de un producto software.	Implantar el proceso de gestión de configuración	ingenieros, gestores

Tabla E-2. REFSENO: Ejemplo de glosario de conceptos.

Concepto: Artefacto									
Super-concepto: CONCEPT									
Capa	Nombre	Descripción	Cardinalidad	Tipo	Valor defecto	Obligatoriedad	Valor inferido	Atributos inferidos	Peso estándar
artef	calidad	Medida cualitativa de la calidad, especialmente de la documentación existente sobre el artefacto.	1	calidad	-	Sí	-	-	1
	tipo	Naturaleza del artefacto.	1..*	tipoA	-	Sí	-	-	1
	edad	Número de años desde que se obtuvo la primera versión.	1	entero	-	No	-	-	1
ctxt	producto	Cada artefacto pertenece a un producto software	1	incluye-a [Producto]. [artefactos]	-	Sí	-	-	5
	versiones	Lista de versiones del artefacto.	1..*	es-parte-de [Versión]. [artefactos]	-	No	-	-	1
	compuesto por	Artefactos más simples que lo forman.	0..*	incluye-a [Artefactos]. [componente de]	-	No	-	-	1
	componente de	Artefactos más complejos de los que forma parte.	0..*	es-parte-de [Artefactos]. [compuesto por]	-	No	-	-	1
sim-artef, sim-i/f, sim-ctxt: estándar precondición: TRUE aserción: TRUE									

Tabla E-3. REFSENO: Ejemplo de una tabla de atributos de concepto: Artefacto.

Concepto: Producto									
Super-concepto: CONCEPT									
Capa	Nombre	Descripción	Cardi- nalidad	Tipo	Valor defecto	Obliga- toriedad	Valor inferido	Atributos inferidos	Peso estándar
artef	madurez	Etapas en el ciclo de vida según el modelo de Rajlich y Bennett (2000).	1	madurezP	“inicial”	Sí	-	-	1
	tamaño	Medida cualitativa del tamaño	1	tamañoP	-	No	-	-	1
	composición	Nivel de abstracción de los artefactos que lo forman.	1	compoP	-	Sí	-	-	1
	tipo de aplicación	Las características del mantenimiento están muy influidas por ella.	1	tipoP	-	Sí	-	-	1
	calidad	Medida cualitativa de la calidad, especialmente de la documentación.	1	calidad	-	Sí	-	-	1
	edad	Número de años desde que se obtuvo la primera versión.	1	entero	-	Sí	-	-	1
	entregable	Indica si el artefacto debe ser entregado al cliente.	1	booleano	-	Sí	-	-	1
ctxt	versiones	Lista de versiones del producto que han sido producidas.	1..*	es-origen-de [Versión]. [producto]	-	No	-	-	2
	artefactos	Lista de artefactos que forman el producto.	1..*	incluye-a [Artefacto]. [producto]	-	Sí	-	-	2
sim-artef, sim-i/f, sim-ctxt: estándar precondición: TRUE aserción: TRUE									

Tabla E-4. REFSENO: Ejemplo de una tabla de atributos de concepto: Producto.

Concepto: Versión									
Super-concepto: CONCEPT									
Capa	Nombre	Descripción	Cardi- nalidad	Tipo	Valor defecto	Obliga- toriedad	Valor inferido	Atributos inferidos	Peso estándar
ctxt	producto	Cada versión lo es de un producto software.	1	su-origen-es [Producto]. [versiones]	-	Sí	-	-	5
	artefactos	Lista de artefactos que están contenidos en la versión.	1..*	incluye-a [Artefacto]. [versiones]	-	Sí	-	-	3
sim-artef, sim-i/f, sim-ctxt: estándar precondición: TRUE aserción: TRUE									

Tabla E-5. REFSENO: Ejemplo de una tabla de atributos de concepto: Versión.

El diagrama de la figura siguiente es la representación alternativa propuesta por REFSENO para los atributos no terminales.

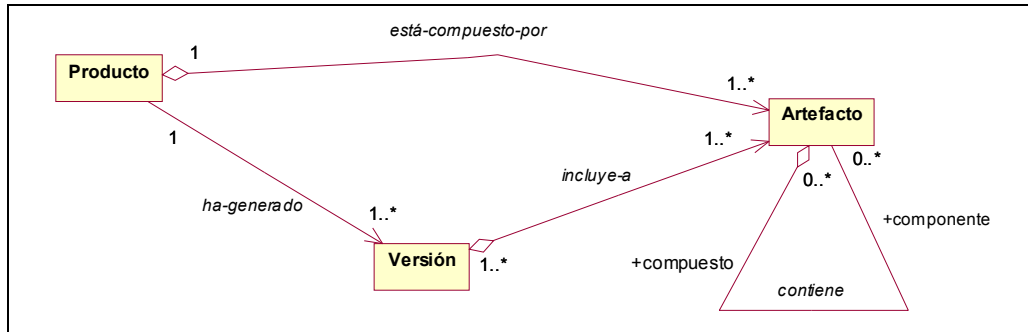


Figura E-1. REFSENO: Ejemplo de representación alternativa de atributos no terminales.

A continuación se incluyen las tablas de tipos de atributos terminales y de clases de atributos no terminales, y un esquema de la tabla de símbolos.

Nombre	Supertipo	Rango valores	Unidad medida	Similaridad
Calidad	Símbolo ordenado	“muy baja”, “baja”, “media”, “alta”, “muy alta”	N/a	
TipoA	Símbolo	“documento”, “módulo”, “componente”, “archivo”, ...	N/a	
MadurezP	Símbolo ordenado	“inicial”, “evolución”, “servicio”, “retirada”, “cierre”	N/a	
TamañoP	Símbolo ordenado	“muy pequeño”, “pequeño”, “mediano”, “grande”, “muy grande”	N/a	
TipoP	Símbolo	“gestión”, “científica”, “específica”, “empotrada”, ...	N/a	
compoP	Símbolo	“caja negra”, “abiertos”	N/a	

Tabla E-6. REFSENO: Ejemplo de una tabla de tipos.

Clase	Nombre inverso	Propósito	Estructura	Propiedades
Es-origen-de	Su-origen-es	Representa que una instancia se origina como consecuencia y después de la existencia de la otra.	Conjunto de árboles	transitividad, dependencia existencial

Tabla E-7. REFSENO: Ejemplo de una tabla de clases de atributos no terminales.

Tipo	Símbolo	Descripción
....		
	documento	documento escrito
	módulo	módulo de código fuente
TipoA	componente	Componente ejecutable (COTS, activos, JavaBean, etc.)
	archivo	Archivos electrónicos distintos de los anteriores símbolos (por ejemplo, archivo HTML).
....		

Tabla E-8. REFSENO: Extracto de un glosario de símbolos de ejemplo.

F. SPEM: Software Process Engineering Metamodel.

SPEM (*Software Process Engineering Metamodel*) es una especificación de OMG (2002d) en fase de aprobación. Este anexo está basado en la versión no oficial publicada en mayo de 2002 ya que su aprobación formal todavía no se ha producido, aunque estaba prevista para julio de 2002.

F.1. Características Generales.

SPEM describe un metamodelo genérico para la descripción de procesos software concretos que está basado en MOF (ver anexo C) y utiliza UML como notación de modelado. Por tanto, se basa en los principios de orientación a objetos. En esta propuesta no está incluida la reificación de los procesos, es decir, la planificación y ejecución de proyectos usando un modelo de proceso descrito con SPEM.

El metamodelo SPEM sirve como plantilla para la creación de modelos de procesos concretos, como podrían ser el “Proceso Unificado de Desarrollo de software de Rational” (RUP), el modelo de evaluación y mejora de procesos de ISO 15504, o el proceso de mantenimiento definido en la metodología MANTEMA (ver anexo B). Por tanto, SPEM es un metamodelo del nivel M2 de MOF, mientras que estos procesos citados se definirían en base a SPEM en el nivel M1.

Además del metamodelo, la especificación SPEM también está estructurada como un perfil UML (*UML profile*), es decir, una variante de UML que utiliza los mecanismos de extensión de UML en una forma estándar para un propósito particular. Esto permite intercambiar definiciones de procesos, tanto con herramientas basadas en MOF, como con las basadas en UML.

SPEM especifica el conjunto mínimo de elementos necesarios para describir cualquier proceso software concreto, sin incluir constructores para áreas o disciplinas específicas; de forma que en SPEM se describe un metamodelo genérico. El objetivo fundamental de esta especificación es tratar de homogeneizar la diversidad terminológica existente en los lenguajes de modelado de procesos software (ver capítulo 3), en los que los mismos conceptos se tratan con nombres diferentes.

La especificación SPEM está compuesta por un conjunto de paquetes en los que se describen cada uno de sus elementos. Todos estos paquetes se construyen a partir del paquete SPEM_Foundation, que es un subconjunto de UML 1.4 (OMG, 2001a), y el paquete de extensiones de SPEM (SPEM_Extensions_Package), que añade los constructores y la semántica necesaria para la ingeniería del proceso software. El paquete SPEM_Foundation está formado, a su vez, por cinco sub-paquetes:

- “*Data Types*”, subconjunto del equivalente en UML;
- “*Core*”, similar al equivalente de UML pero eliminando algunas clases y atributos y con algunos cambios, por ejemplo, haciendo abstractas algunas clases;
- “*Actions*”, que es un subconjunto del paquete Common_Behavior de UML;
- “*State Machines*”, un subconjunto del de mismo nombre en UML ; y

- “*Model Management*”, también un subconjunto del equivalente en UML.

El modelo conceptual de SPEM está basado en la idea de que un proceso software consiste en la colaboración entre entidades abstractas y activas denominadas “roles de proceso” (*process roles*) que realizan operaciones denominadas “actividades” (*activities*) sobre entidades tangibles denominadas “productos de trabajo” (*work products*). El fundamento de los procesos software consiste en la interacción o colaboración de múltiples roles mediante el intercambio de productos de trabajo y la ejecución o disparo de ciertas actividades. El objetivo de un proceso es llevar a un estado bien definido a un conjunto de productos de trabajo. En la Figura F-1 se representa en UML este modelo conceptual básico. Esta representación es directamente traducible y representable en el modelo MOF.

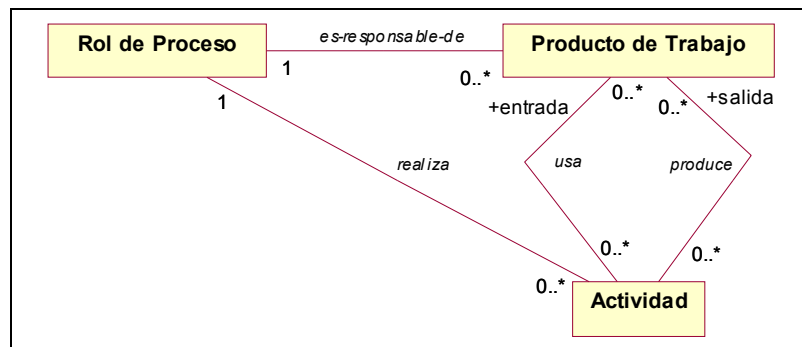


Figura F-1. Modelo conceptual básico de SPEM.

Debido a que el Entorno MANTIS está basado en la arquitectura multicapa de MOF, a continuación presentamos con más detalle el metamodelo SPEM basado en MOF, y después, de forma más resumida, el perfil UML SPEM.

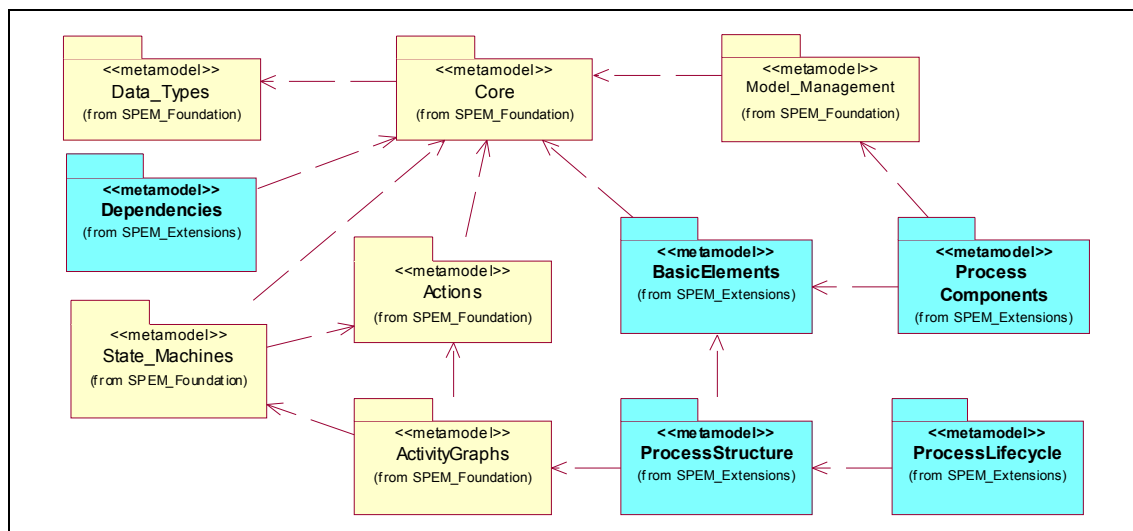


Figura F-2. Estructura de paquetes de SPEM.

F.2. Metamodelo.

El metamodelo SPEM se estructura básicamente en 5 paquetes: Elementos Básicos (*Basic Elements*), Dependencias (*Dependences*), Estructura del Proceso (*Process Structure*), Componentes del Proceso (*Process Components*), y Ciclo de Vida del Proceso (*Process Lifecycle*). En la Figura F-2 se presenta la estructura completa de paquetes de SPEM, con los tres sub-paquetes del SPEM_Foundation más los cinco citados.

A continuación se describe con más detalle el contenido de los cinco paquetes básicos que componen el metamodelo.

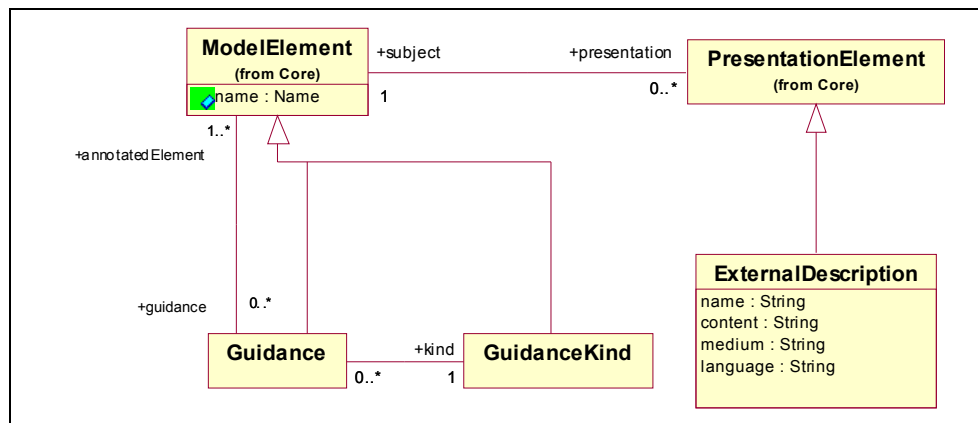


Figura F-3. SPEM: paquete "Elementos Básicos".

F.2.1. Paquete Elementos Básicos.

En este paquete se describen los elementos básicos necesarios para la descripción de procesos (Figura F-3). Estos elementos son:

- Descripción Externa (*ExternalDescription*). Una descripción de los elementos del modelo de forma que sea comprensible para el lector. Cada elemento del modelo está asociado con una o más descripciones externas. Esta clase está compuesta por los siguientes atributos:
 - Contenido (*content*). Descripción en lenguaje natural del elemento.
 - Nombre (*name*). Nombre del elemento del modelo en lenguaje natural.
 - Lenguaje (*language*). Nombre del idioma usado para los valores de los atributos contenido y nombre.
 - Medio (*medium*). Descripción del medio y formato de la descripción externa.
- Guía (*Guidance*). Cada elemento del modelo puede tener asociadas cero o más guías, las cuales proporcionan información más detallada para los encargados del proceso. Cada guía se asocia con un determinado Tipo de Guía (*GuidanceKind*), cuyo nombre indica que clase de guía es. Es posible añadir nuevos tipos de guía si así se considera oportuno. Los elementos Tipo de Guía enumerados en la especificación son:

- Técnica (*technique*). Es un algoritmo preciso que es usado para crear un producto de trabajo (workproduct).
- Perfil UML (*UML profile*). Proporciona mecanismos que especializan UML para un entorno específico, como por ejemplo para C++, JAVA, CORBA o un propósito específico, como análisis, diseño, etc. De esta forma se dispone de una semántica basada en UML para dominios concretos.
- Lista de Comprobación (*checklist*). Una lista de comprobación es un documento que representa una lista de elementos que se deben completar.
- Tutor de herramienta (*tool mentor*). Muestra como usar una herramienta específica para realizar una actividad; por ejemplo, cómo usar Rational Rose para especificar casos de uso en el modelo de desarrollo RUP.
- Línea Guía (*guideline* o *script*). Es un conjunto de reglas y recomendaciones sobre cuál debe ser el aspecto o cómo se debe organizar un determinado producto de trabajo.
- Plantilla (*template*). Es un documento predefinido que proporciona un formato estándar para un producto de trabajo concreto.
- Estimación (*estimate*). Describe el esfuerzo asociado con un elemento en particular

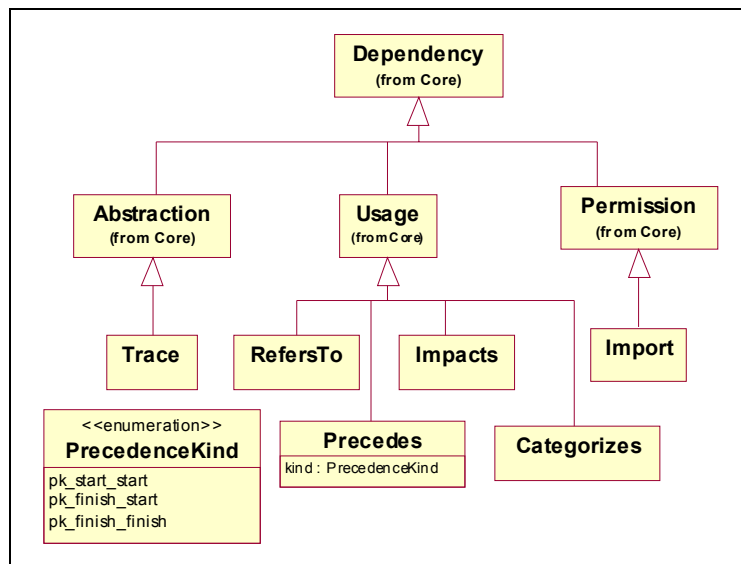


Figura F-4. SPEM: paquete "Dependencias".

F.2.2. Paquete Dependencias.

En este paquete se incluyen los tipos de dependencias definidos en SPEM (Figura F-4) para ingeniería de procesos:

- Categoriza (*Categorizes*). Es una relación que va de un Paquete determinado a un elemento de proceso de otro paquete, y proporciona un medio para asociar los elementos de proceso con múltiples categorías.
- Impacta (*Impacts*). Esta dependencia relaciona productos de trabajo, indicando que la modificación de uno podría invalidar el otro.

- Importa (*Import*). Esta dependencia representa que los contenidos de un paquete destino se añaden al espacio de nombres (*namespace*) de un paquete origen. Tiene la misma semántica que “UML import” salvo que en SPEM todos los elementos tienen visibilidad pública.
- Precede (*Precedes*). Es una relación entre actividades, o entre definiciones de trabajo (*WorkDefinition*), indicando dependencias “comenzar-comenzar”, “acabar-comenzar” o “acabar-acabar” entre estos elementos, en función del valor de su atributo clase “*kind*”.
- Refiere a (*RefersTo*). Relaciona dos elementos de proceso para asegurar que están incluidos en el mismo componente de proceso (ver apartado F.2.4). Se aplica cuando el texto de un elemento de proceso hace referencia, por nombre o contenido, a otro elemento.
- Traza (*Trace*). Es una relación entre definiciones de trabajo o elementos de información (*InformationElements*) usada para la traza (seguimiento) de requisitos y cambios en los modelos. Tiene la misma semántica que “UML trace”.

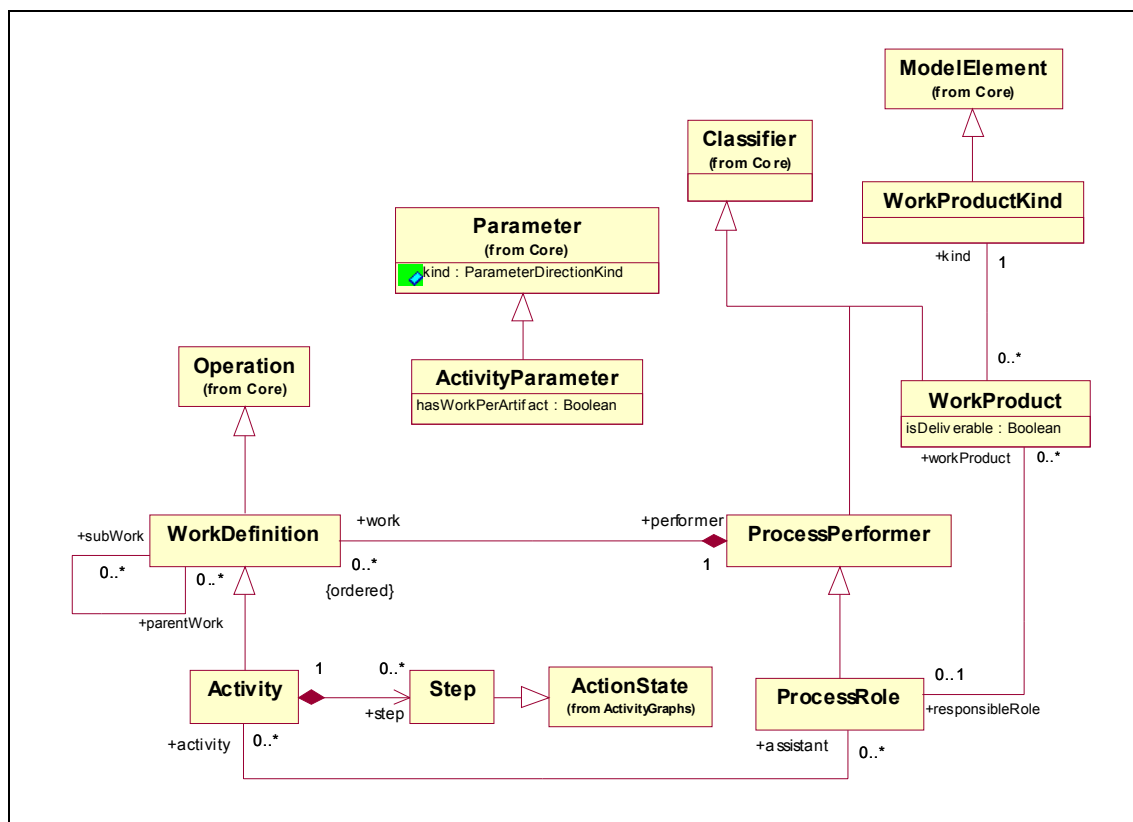


Figura F-5. SPEM: paquete "Estructura del Proceso".

F.2.3. Paquete Estructura del Proceso.

Este paquete incluye los principales elementos estructurales a partir de los cuales se construye la descripción de un proceso (Figura F-5):

- Producto de Trabajo (*WorkProduct*) y Categoría de Producto de Trabajo (*WorkProduct Kind*). Un producto de trabajo o artefacto es cualquier cosa que es producida, consumida o modificada por un proceso. Podría ser un fragmento de información, un documento, un

modelo, código fuente, etc.. Describe una clase de producto de trabajo. En cambio, una categoría de producto de trabajo describe una generalización de producto de trabajo como, por ejemplo, documento de texto, modelo UML, o ejecutable. El atributo de producto de trabajo “*isDeliverable*” indica si el producto es un entregable formal del proceso. Entre las asociaciones de producto de trabajo destacan las siguientes:

- Es una especialización de “*Classifier*”. De esta forma, puede participar en asociaciones y contener definiciones anidadas pero no posee características (*Features*).
 - Pueden describirse productos de trabajo formados por agregación de otros más simples. Por ejemplo, un plan de desarrollo software contiene un plan de personal, un plan de gestión de la configuración, etc.
 - Puede estar asociado con un rol (*responsibleRol*) que es el responsable de su producción.
 - Debe estar asociado con una categoría de producto de trabajo.
 - Puede estar asociado con una máquina de estados (*StateMachine*) que describe los posibles estados en los que puede estar el producto de trabajo y las transiciones posibles entre dichos estados.
- Definición de Trabajo (*WorkDefinition*) y Parámetro de Actividad (*ActivityParameter*). Definición de trabajo es una clase no abstracta de operación que describe el trabajo realizado en el proceso. La principal clase en la que se especializa es Actividad (*Activity*), pero también tiene otras especializaciones (Ciclo de Vida, Fase, Iteración) que pertenecen al paquete del Ciclo de Vida del Proceso. Tiene entradas y salidas explícitas referidas vía parámetro de actividad. El atributo “*kind*” de la clase Parámetro se usa para indicar si el producto de trabajo asociado es de entrada, salida, de entrada modificable, o un valor devuelto. El atributo “*Tiene Trabajo por Artefacto*” (*hasWorkPerArtifact*) señala si son necesarias instancias múltiples de la clase definición de trabajo para disponer de una por instancia del correspondiente producto de trabajo. Por ejemplo, la definición de trabajo “Escribir código” podría tener como entradas los productos de trabajo “Estándar de Codificación” y “Clase”, interesando su réplica (instancias múltiples) una vez por cada clase pero no por cada estándar de codificación. Entre las asociaciones de la clase definición de trabajo destacan las siguientes:
 - Puede estar compuesta por otras más simples (subtrabajos, *subworks*).
 - Está relacionada con los productos de trabajo que se usan a través de la clase parámetro de actividad, que indica si los productos se están usando como entradas o como salidas mediante el atributo “*kind*”. La definición de trabajo usa los productos de entrada y crea o modifica los productos de salida.
 - El propietario de una definición de trabajo es la clase Realizador del Proceso (*ProcessPerformer*), que representa el rol principal encargado de la realización de esa definición de trabajo.
 - Puede ser referenciada por un elemento EstadoAcción (*ActionState*) desde un grafo de actividad (*ActivityGraph*).
 - Actividad (*Activity*) y Paso (*Step*). Actividad es la subclase principal de definición de trabajo y describe una parte de trabajo realizado por un rol de proceso (tareas, operaciones o acciones que un determinado rol realiza o asiste). Una actividad puede estar formada por un conjunto de elementos atómicos denominados pasos.

- Realizador del Proceso (*ProcessPerformer*) y Rol del Proceso (*ProcessRole*). La clase realizador del proceso describe el encargado de realizar un conjunto de definiciones de trabajo. Rol del proceso es una subclase de realizador del proceso para describir las responsabilidades asociadas a los productos de trabajo, definiendo los roles que realizan y asisten en actividades específicas. Entre las asociaciones de estas clases cabe destacar las siguientes:
 - Realizador del proceso es una subclase de “*Classifier*”, por lo que puede participar en asociaciones y relaciones de herencia dentro de la definición de proceso.
 - Un rol de proceso es responsable de un conjunto de productos de trabajo.
 - Un rol de proceso es el realizador de un conjunto de actividades.
 - Un realizador del proceso es el realizador de definiciones de trabajo agregadas de alto nivel que no se pueden asociar con roles de proceso individuales.

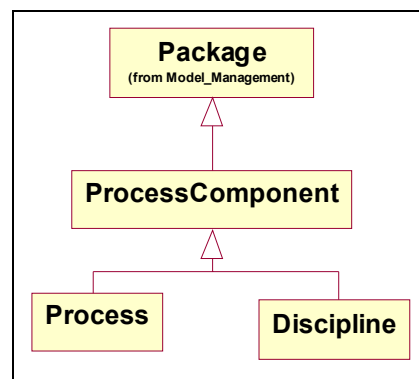


Figura F-6. SPEM: paquete "Componentes del Proceso".

F.2.4. Paquete Componentes del Proceso.

Este paquete contiene los elementos necesarios para dividir una o más descripciones de proceso en partes auto-contenidas sobre las que se pueden aplicar procesos de gestión de la configuración o control de versiones (Figura F-6). Los elementos representados son:

- Paquete (*Package*). En SPEM, al igual que en UML, un paquete es un contenedor que puede tanto poseer como importar cualquier elemento de definición de procesos. Se pueden usar junto con la dependencia Categoriza (*categorizes*) para implementar categorizaciones, de forma que el paquete representa una categoría y los elementos del mismo relacionados con esta dependencia representan la pertenencia a esa categoría, siendo el nombre de la categoría el nombre del paquete.
- Componente del Proceso (*ProcessComponent*). Es una parte de una descripción de un proceso que es internamente consistente y podría reutilizarse junto con otros componentes de proceso para constituir un proceso completo. Importa un conjunto no arbitrario de elementos de definición de procesos modelados en SPEM como Elementos del Modelo (*ModelElements*). Dicho conjunto debe ser auto-contenido, lo que significa que no puede haber dependencias de tipo “referencia_a” a elementos que están fuera del componente.

Además, debe ser internamente consistente, lo que implica que las multiplicidades y restricciones definidas para el metamodelo como un todo deben ser satisfechas dentro del alcance del componente.

- Proceso (*Process*). Es un componente de proceso que se diferencia de los demás en que no está incluido en otros componentes. Es el elemento raíz en un modelo de proceso. Tiene asociado un ciclo de vida (ver apartado F.2.5).
- Disciplina (*Discipline*). Es una especialización de paquete que particiona las actividades de un proceso de acuerdo con un tema común determinado. Esta partición implica que las guías asociadas y los productos de trabajo de salida son categorizados de la misma forma bajo el mismo tema. La inclusión de una actividad en una disciplina se representa por la dependencia categoriza con la restricción adicional de que cada actividad es categorizada por exactamente una disciplina. Ejemplos de disciplinas en el “Proceso Unificado de Rational” son: Gestión de Requisitos, Diseño, Pruebas, Gestión de Proyectos, etc.

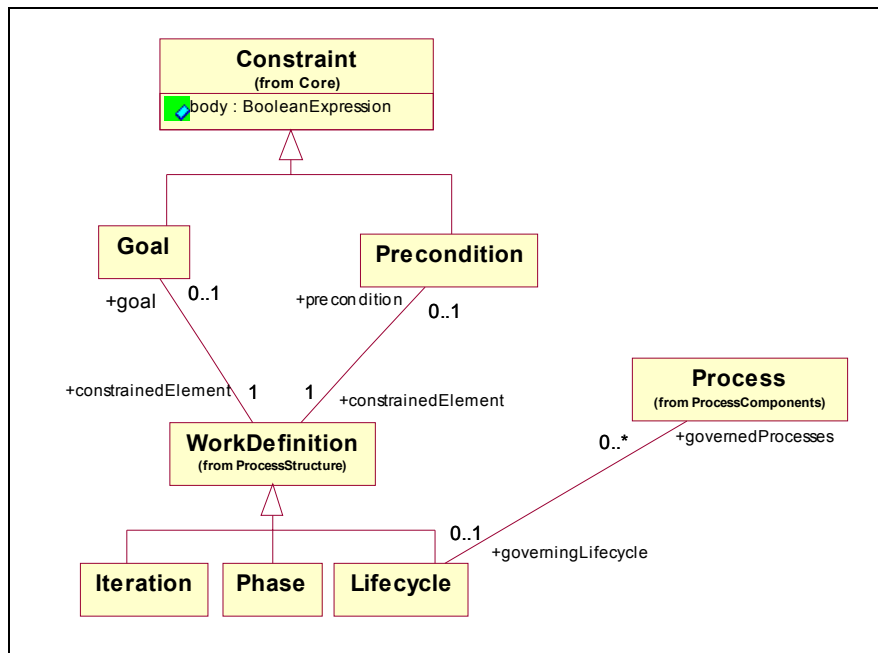


Figura F-7. SPEM: paquete "Ciclo de Vida del Proceso".

F.2.5. Paquete Ciclo de Vida del Proceso.

En este paquete se incluyen los elementos de definición de procesos que ayudan a definir cómo se van a ejecutar los procesos. Describen o restringen el comportamiento del proceso a realizar, y se utilizan para asistir en la planificación, ejecución y monitorización del proceso. Con estos elementos se establece el orden de ejecución del proceso permitiendo definir también las iteraciones y fases del mismo. Estos elementos no describen la animación o reificación del proceso en sí misma, sino que son elementos de descripción que ayudan a la planificación y ejecución del proceso descrito. En la Figura F-7 están representados los siguientes elementos:

- Fase (*Phase*). Es una especialización de la clase definición de trabajo tal que su precondition define el criterio de entrada a la fase y su objetivo (frecuentemente conocido como hito o “*milestone*”) define el criterio de salida de la fase. Las fases se definen con la restricción adicional de secuencialidad, es decir, sus reificaciones se ejecutan bajo una serie de hitos y hay un solapamiento mínimo (o ninguno) de las fases a lo largo del tiempo.
- Ciclo de Vida (*LyfeCycle*). Un ciclo de vida de un proceso se define como una secuencia de fases que cumplen con un determinado objetivo. Se asocia con un conjunto de fases mediante el uso de la asociación “*subwork*” y se asocia con uno o más procesos mediante la asociación “*governedProcesses*”.
- Iteración (*Iteration*). Es una definición de trabajo cuyo hito es de menor importancia que los anteriores.
- Precondición (*Precondition*) y Objetivo (*Goal*). A cada definición de trabajo se puede asociar una precondition y un objetivo que son restricciones expresadas en forma booleana (en formato texto) similares a una condición de salvaguarda en UML. Esta condición se refieren a los estados posibles de los productos de trabajo.

F.3. SPEM como Perfil UML.

Como perfil UML, SPEM define capacidades de modelado destinadas al dominio concreto de los procesos software aprovechando las facilidades de expresividad que proporciona UML. Por ejemplo, el modelado de casos de uso, que en ocasiones se usa para modelar procesos, no está definido como una facilidad específica de SPEM, pero puede ser heredada de UML. En general, el uso de UML como base para el modelado de procesos aprovecha las ventajas en cuanto al posible alineamiento con otros lenguajes de modelado. Además, UML cuenta con tecnología de soporte (herramientas CASE variadas) que también puede ser aprovechada en el dominio de los procesos software.

Los pasos necesarios para definir un perfil UML para SPEM son los siguientes:

1. Identificar qué subconjunto de las clases del metamodelo UML van a incluirse en el perfil.
2. Para la mayoría de las clases del metamodelo SPEM, identificar una “clase base” del subconjunto de UML que, estereotipada de forma apropiada, debe actuar en lugar de la clase SPEM.
3. Definir la forma de emular cada atributo y asociación en SPEM. En el perfil SPEM los atributos se emulan mediante el uso de valores etiquetados (*tagged values*). La mayoría de las asociaciones son muy análogas al metamodelo UML.
4. Para las partes del subconjunto de UML que tienen una posible correspondencia con conceptos de SPEM, pero que no se utilizan directamente para emular el metamodelo SPEM, mostrar como se corresponden con los conceptos de SPEM. En SPEM esto se aplica en el uso de “Diagramas de Casos de Uso”, “Diagramas de Actividad” y “Máquinas de Estados”.

5. Proporcionar restricciones adicionales sobre el metamodelo UML.
6. Definir iconos que denoten los conceptos de SPEM que son representados por estereotipos UML.

A modo de resumen del resultado de llevar a cabo dichos pasos, en la Figura F-8 se han representado las correspondencias entre los conceptos de SPEM y las clases base de UML.

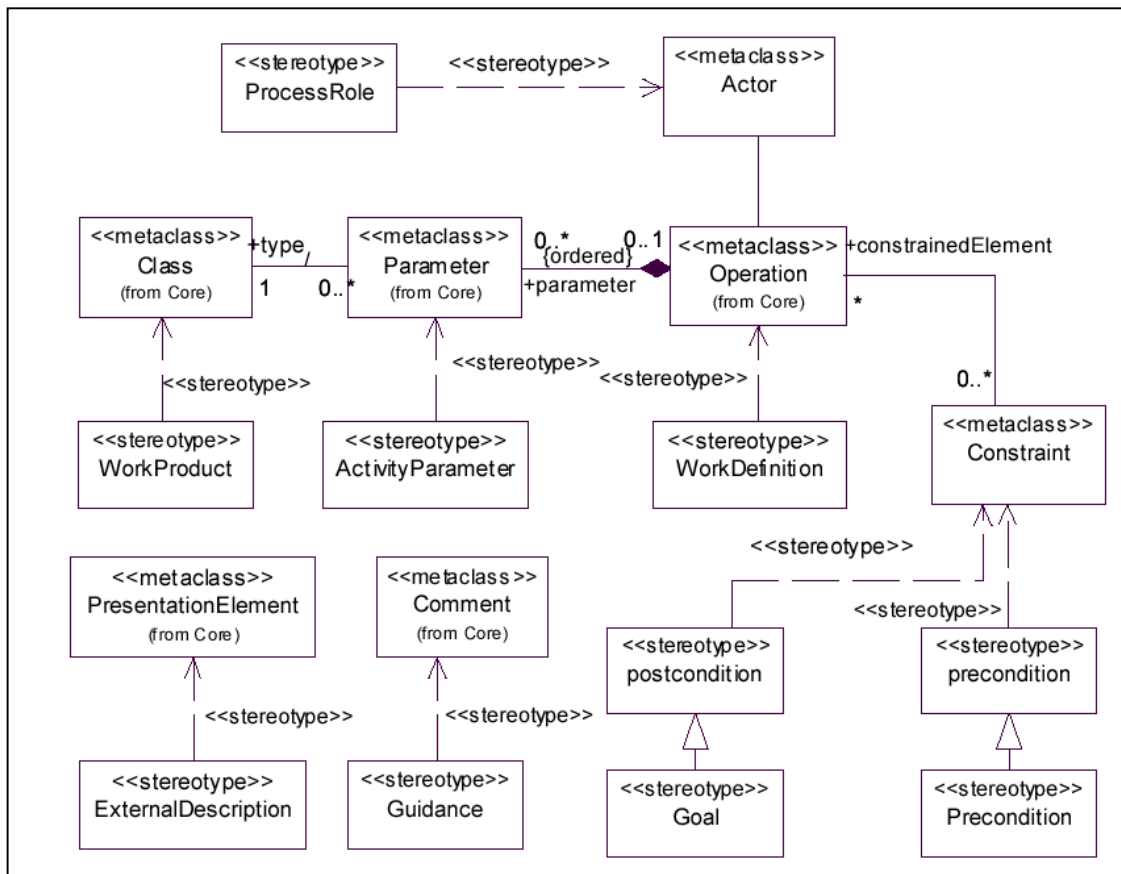


Figura F-8. Correspondencias simples entre el perfil SPEM y las clases base de UML.

Adicionalmente, los atributos de SPEM se representan como valores etiquetados UML y las asociaciones se representan de varias formas, entre las que cabe destacar las siguientes:

- *Guía:Elemento_de_Notación* se representa como la asociación UML *Comentario::Elemento_de_Notación*.
- *Parámetro_de_Actividad::Tipo* se representa como *Parámetro::Tipo*.
- *Definición_de_Trabajo::Propietario* se representa como *Característica::Propietario*.
- *Definición_de_Trabajo::SubTrabajo* se representa de forma indirecta usando grafos de actividades.
- *Actividad::Pasos* también se representa como grafos de actividades.
- *Definición_de_Trabajo::Objetivo* o *Definición_de_Trabajo::Precondición* se representan como *Elemento_del_Modelo::Restricción*.

- *Proceso::Ciclo_de_Vida* se representa con un nuevo estereotipo de “Abstracción” denominado <<gobierna>>, que relaciona los procesos con sus ciclos de vida.

G. Correspondencias entre los Niveles Conceptuales de MANTIS.

En este anexo se presentan varios ejemplos de correspondencias entre los niveles M3, M2 y M1 de la arquitectura conceptual de MANTIS, explicada en el capítulo 5. No se ha pretendido ser exhaustivo (lo que sería prácticamente imposible por razones de espacio) sino que se han buscado ejemplos que desarrollen algunos de los aspectos más importantes de las ontologías y metamodelos del Entorno, y cómo la arquitectura conceptual permite representar y trabajar con ellos de forma más organizada y sistemática.

G.1. Correspondencias M3-M2.

En este apartado vamos a mostrar algunas correspondencias que se definen entre el modelo MOF (anexo C), perteneciente al nivel M3, y el paquete básico del metamodelo genérico de proceso software (capítulo 5), perteneciente al nivel M2.

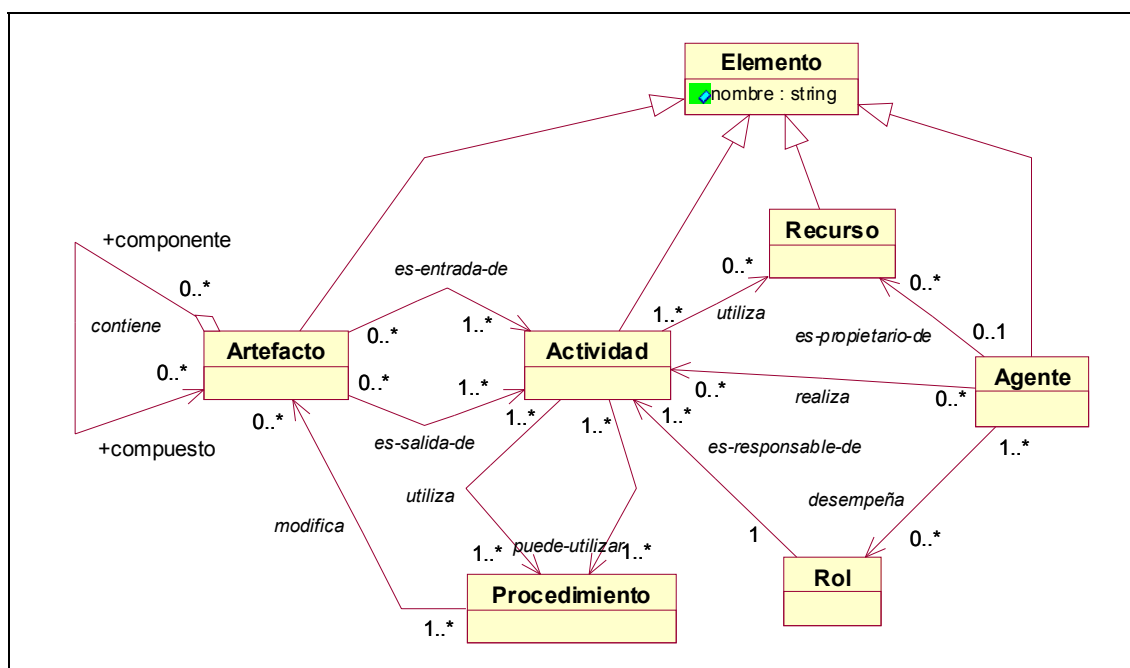


Figura G-1. Paquete básico adaptado del metamodelo de proceso software.

Con fines didácticos, en la Figura G-1 hemos realizado tres retoques respecto del paquete básico definido en el capítulo 5:

- hemos eliminado las clases “Proyecto” y “Restricción” para así poder centrarnos en los elementos de un proyecto;

- no hemos considerado los atributos salvo uno (llamado nombre) que hemos incluido en la clase “Elemento” y que será heredado por las clases que se especializan de ella (“Actividad”, “Recurso”, etc.); y
- hemos incluido la asociación “realiza” entre las clases “Agente” y “Actividad”, aunque realmente pertenece al paquete de los flujos de trabajo.

La Tabla G-1 muestra las correspondencias entre los constructores básicos de MOF y los elementos mostrados en la figura anterior. Se han incluido abreviaturas para simplificar la escritura posterior. Los documentos XMI y DTD referidos a este ejemplo se pueden consultar en el apartado I.1.

Clases M3	Clases M2 (ejemplares M3)	
Clase-MOF	Act	Actividad
	Age	Agente
	Art	Artefacto
	Ele	Elemento
	Pro	Procedimiento
	Rec	Recurso
	Rol	Rol
Atributo-MOF	Nombre	Clase: Elemento
Asociación-MOF	Contiene-Art-Art	Artefacto contiene Artefacto
	Desempeña-Age-Rol	Agente desempeña Rol
	EsEntradaDe-Art-Act	Artefacto es entrada de Actividad
	EsPropietarioDe-Age-Rec	Agente es propietario de Recurso
	EsResponsableDe-Rol-Act	Rol es responsable de Actividad
	EsSalidaDe-Art-Act	Artefacto es salida de Actividad
	EsUn-Act-Ele	Actividad es un Elemento
	EsUn-Age-Ele	Agente es un Elemento
	EsUn-Art-Ele	Artefacto es un Elemento
	EsUn-Rec-Ele	Recurso es un Elemento
	Modifica-Pro-Art	Procedimiento modifica Artefacto
	PuedeUtilizar-Act-Pro	Actividad puede utilizar Procedimiento
	Realiza-Age-Act	Agente realiza actividad
	Utiliza-Act-Pro	Actividad utiliza Procedimiento
	Utiliza-Act-Rec	Actividad utiliza Recurso

Tabla G-1. Ejemplos de correspondencias M3-M2.

G.2. Correspondencias M2-M1.

La principal información manejada al nivel M1 en el entorno MANTIS son los modelos concretos de proceso y entre ellos, el más significativo es el modelo de proceso de mantenimiento definido por la metodología MANTEMA (ver anexo B). Por razones didácticas hemos elegido para mostrar en este apartado las correspondencias entre el paquete indicado en el apartado anterior y el grupo de actividades de MANTEMA conocido como “Actividades del mantenimiento no planificable”, es decir, las actividades centrales que se llevan a cabo cuando se producen peticiones de mantenimiento correctivo urgente. En la Figura G-2 se muestra esta parte del modelo MANTEMA utilizando la extensión SPEM de UML (ver anexo F).

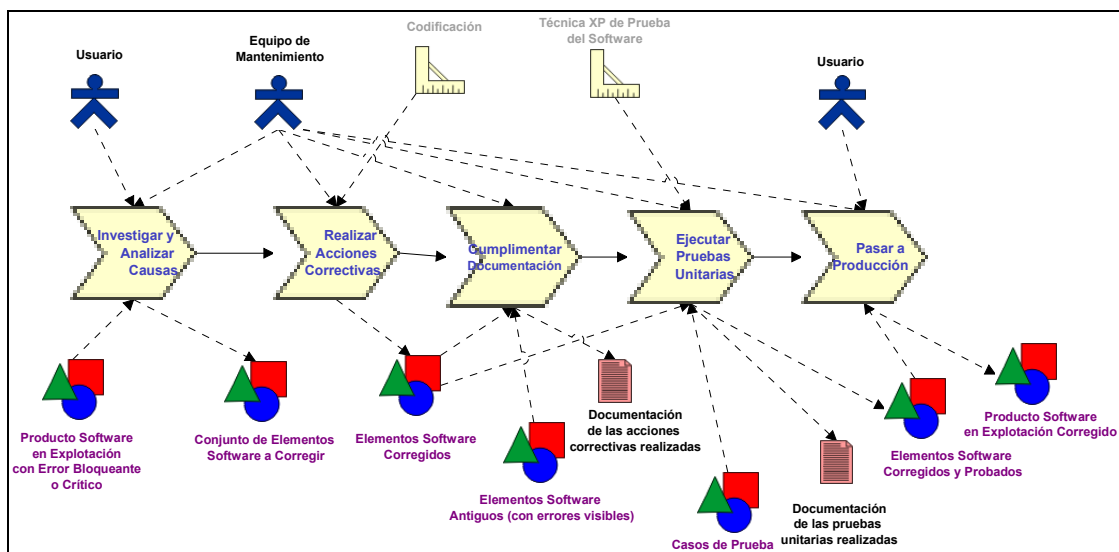


Figura G-2. Representación SPEM del mantenimiento no planificable de MANTEMA.

En la Tabla G-2 se muestran algunas de las correspondencias entre las clases del nivel M2 y las del nivel M1 que actúan como ejemplares de las anteriores. Los documentos XMI y DTD referidos a este ejemplo se pueden consultar en el apartado I.2.

Clases M2	Clases M1 (ejemplares M2)
Act Actividad	Análisis del error (NP1)
	Cierre intervención (NP3)
	Cumplimentar documentación (NP2.2)
	Ejecutar pruebas unitarias (NP2.3)
	Intervención correctiva urgente (NP2)
	Investigar y analizar causas (NP 1.1)
	Pasar a producción (NP3.1)
	Realizar acciones correctivas (NP2.1)
Age Agente	Mantenedor
	Usuario
Art Artefacto	Caso de prueba
	Conjunto de elementos software a corregir
	Documentación de la acción correctiva realizada
	Documentación de las pruebas unitarias realizadas
	Elemento software antiguo
	Elemento software corregido
	Elemento software corregido y probado
	Producto software en explotación con error bloqueante o crítico
Contiene-Art-Art Artefacto contiene Artefacto	“Producto software” contiene “Documentación”
	“Mantenedor” desempeña “Equipo de mantenimiento”
Desempeña-Age-Rol Agente desempeña Rol	“Usuario” desempeña “Usuario”

Clases M2	Clases M1 (ejemplares M2)
EsEntradaDe-Art-Act Artefacto es entrada de Actividad	“Elemento software a corregir” es entrada de “Realizar acciones correctivas”
	“Elemento software antiguo” es entrada de “Cumplimentar documentación”
	“Elemento software corregido y probado” es entrada de “Pasar a producción”
	“Elemento software corregido” es entrada de “Cumplimentar documentación”
	“Elemento software corregido” es entrada de “Ejecutar Pruebas Unitarias”
	“Producto software en explotación con error bloqueante o crítico” es entrada de “Investigar y analizar causas”
EsPropietarioDe-Age-Rec Agente es propietario de Recurso	“Mantenedor” es propietario de “Computadora”
EsResponsableDe-Rol-Act Rol es responsable de Actividad	“Equipo de mantenimiento” es responsable de “Ejecutar pruebas unitarias”
	“Equipo de mantenimiento” es responsable de “Investigar y analizar causas”
	“Equipo de mantenimiento” es responsable de “Pasar a producción”
	“Equipo de mantenimiento” es responsable de “Realizar acciones correctivas”
	“Usuario” es responsable de “Investigar y analizar causas”
EsSalidaDe-Art-Act Artefacto es salida de Actividad	“Documentación con las pruebas unitarias realizadas” es salida de “Ejecutar pruebas unitarias”
	“Documentación de las acciones correctivas realizadas” es salida de “Cumplimentar Documentación”
	“Elemento software a corregir” es salida de “Investigar y analizar causas”
	“Elemento software corregido y probado” es salida de “Ejecutar pruebas unitarias”
	“Elemento software corregido” es salida de “Realizar acciones correctivas”
	“Producto software en explotación corregido” es salida de “Pasar a producción”
Modifica-Pro-Art Procedimiento modifica Artefacto	“Codificación” modifica “Código fuente”
Pro Procedimiento	Codificación
	Técnica XP de pruebas del software
PuedeUtilizar-Act-Pro Actividad puede utilizar Procedimiento	“Ejecutar pruebas unitarias” puede utilizar “Técnica XP de prueba del software”
	“Realizar acciones correctivas” puede utilizar “Codificación”
Realiza-Age-Act Agente realiza actividad	“Mantenedor” realiza “Cumplimentar documentación”
Rec Recurso	Computadora
	Herramienta CASE de pruebas
Rol Rol	Equipo de Mantenimiento
	Usuario
Utiliza-Act-Pro Actividad utiliza Procedimiento	“Realizar acciones correctivas” utiliza “Codificación”
Utiliza-Act-Rec Actividad utiliza Recurso	“Realizar acciones correctivas” utiliza “Computadora”

Tabla G-2. Ejemplos de correspondencias M2-M1.

H. Manuales de Usuario.

En este anexo se han incluido resúmenes de los manuales de usuario de algunos de los prototipos desarrollados para el Entorno MANTIS. No se ha tratado de ser exhaustivo y tan solo se ha pretendido que el lector pueda hacerse una mejor idea de su utilidad viendo la manera de utilizarlos. Los manuales que se incluyen pertenecen a las siguientes herramientas:

- El gestor del repositorio, RepManager (apartado H.1);
- El administrador de modelos y metamodelos, METAMOD (apartado H.2); y
- El gestor de recursos humanos en una cartera de proyectos de mantenimiento, CREM (apartado H.3).

H.1. Manual de RepManager.

RepManager no es una aplicación interactiva dirigida a un usuario final, sino que se trata de un componente interno de MANTIS cuyos servicios deben poder ser invocados por cualquier herramienta de metamodelización (HMM) basada en MOF. Por ello, el manual de usuario del componente está dirigido a los programadores de este tipo de aplicaciones que deseen aprovechar este gestor de repositorio para el almacenamiento e importación de metadatos en forma de documentos XMI.

El gestor de repositorio ha sido desarrollado como un componente ActiveX y se puede enlazar con cualquier programa de aplicación. El nombre del componente es "RepManager.dll". La API proporcionada por el componente está dividida, por razones de sencillez, en dos funciones: GenerarDocumentoXMI y CargarDocumentoXMI. A continuación se explica la manera de utilizar cada una.

H.1.1. Servicio GenerarDocumentoXMI.

Esta operación permite que la HMM pueda almacenar sus metadatos en forma de documentos XMI. Devuelve verdadero si la operación se ha realizado correctamente y falso en caso contrario. En el anexo I se muestran ejemplos de documentos XMI y DTD de los niveles M2 y M1 generados automáticamente con este servicio de RepManager.

Los parámetros que hay que suministrar para el correcto funcionamiento de la operación son los siguientes:

Codificación:

Este parámetro, de tipo cadena de texto, permite establecer el formato de codificación del documento XML, que se corresponde con el atributo *encoding* de la etiqueta inicial de un documento XML. Esta etiqueta inicial presenta el siguiente aspecto:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Colección de espacios de nombres:

Es una colección de objetos de tipo ‘*namespace*’. Los XML-namespaces resuelven el problema de conflictos de nombres. Las etiquetas (y atributos) del documento se pueden hacer únicos añadiéndoles un prefijo identificador del *namespace*. Cada *namespace* es identificado por una URL que, como cualquier otra, es única.

Ruta:

Este parámetro es de tipo cadena de texto y especifica la ruta completa (incluido el nombre del documento XML) del archivo XML a generar.

Colección de modelos importados:

Es un parámetro de tipo colección que contiene objetos de tipo ‘Import’ (clase incluida en el componente RepManager). Es posible que el modelo a representar en el documento XMI utilice elementos de otros modelos. En este caso se deben especificar dichos modelos en los elementos de esta colección.

Nombre del DTD a generar:

Es un parámetro de tipo cadena. Si el documento a generar representa un metamodelo, opcionalmente se puede indicar que se genere un DTD correspondiente a dicho metamodelo. Dicho DTD se usa, posteriormente, para validar los documentos XML que representen modelos instanciados a partir de dicho metamodelo. A través de este parámetro se puede especificar cuál es el nombre del DTD a generar. En el caso de que no se desee crear un DTD este parámetro debe ser la cadena vacía.

Modelo o metamodelo:

Este parámetro representa un objeto de tipo ‘Metamodelo’. Esta clase pertenece al componente “Implementación MOF” necesario para la ejecución y comunicación del gestor con las aplicaciones HMM. A través de dicha clase se puede representar o bien un modelo o bien un metamodelo de información según el valor de su propiedad nivel sea 1 o 2 respectivamente. A través de un objeto de este tipo el gestor puede extraer toda la información necesaria para la generación del documento XMI que representa el modelo o metamodelo.

Referencia a DTD:

Parámetro de tipo cadena que permite indicar una referencia a un DTD en la generación del documento. Todo documento XML puede incluir de forma opcional una referencia al DTD que representa su estructura. La línea del documento XML que permite indicar una referencia es de la siguiente forma:

```
<!DOCTYPE XMI SYSTEM 'Model.dtd' >
```

En el caso de generar un metamodelo, por defecto la referencia debe ser al archivo DTD que representa el modelo MOF (único). El gestor de repositorio exporta e importa los metamodelos haciendo uso del DTD de MOF provisto por la empresa Rational como “plugging” de su herramienta CASE “Rose”. En el caso de generar un modelo, la referencia debe ser a un archivo DTD que represente al metamodelo del que el modelo es instancia.

H.1.2. Servicio CargarDocumentoXML.

Este servicio permite que una HMM pueda cargar la información contenida en un documento XMI correspondiente a un modelo o a un metamodelo. Devuelve el resultado verdadero si la operación se ha realizado correctamente y falso en caso contrario. Los parámetros que hay que suministrar para su correcto funcionamiento son los siguientes:

Ruta:

Este parámetro es de tipo cadena de texto y especifica la ruta completa donde se encuentra el fichero XML que representa el modelo o metamodelo a cargar.

Con DTD:

Es un parámetro de tipo *booleano* que especifica si el documento a cargar debe ser validado con respecto al DTD al que hace referencia. En determinados casos puede ser deseable (por razones de eficiencia) que el documento se cargue sin tener en cuenta la validación de su sintaxis con respecto a su DTD. En este caso se debe fijar este parámetro a falso. Cuando el valor de este parámetro es verdadero el documento se carga si respeta la sintaxis de su DTD asociado. En caso contrario (el documento no cumple con las reglas de su DTD) la operación devuelve un error y no se produce dicha carga.

Nivel:

Es un parámetro de tipo entero que especifica si el documento XML a cargar contiene información correspondiente a un metamodelo (valor 2) o a un modelo (valor 1).

Modelo salida:

Este parámetro es de salida y es de tipo 'Metamodelo'. Contiene el modelo o metamodelo resultado de la carga del documento XMI.

Metamodelo:

Este parámetro es opcional y es de tipo 'Metamodelo'. En el caso de que el documento XMI a cargar represente un modelo es necesario indicar en este parámetro el metamodelo del que el modelo es instancia.

H.2. Manual de METAMOD.

A continuación se presenta, de forma breve, el manual de usuario de la herramienta de meta-modelado desarrollada para el Entorno MANTIS. Tal como se comentó en el capítulo 7 y como queda de manifiesto en este manual, esta herramienta no pretende competir con las herramientas CASE comerciales, sino completarlas en aquello que, hasta la fecha, no incluyen: la consulta y gestión de las correspondencias entre modelos y meta-modelos de niveles de abstracción consecutivos (correspondencias M3-M2 o M2-M1 en la arquitectura conceptual de MANTIS).

La interfaz de usuario es bastante similar a interfaces de herramientas CASE muy conocidas, como puede ser Rational Rose. Ello facilita a los usuarios de tales herramientas CASE una interacción más sencilla con METAMOD. Pero además, la herramienta incorpora dos características adicionales con las que no cuentan la mayor parte de herramientas CASE actuales: la posibilidad de definir meta-modelos mediante el uso del lenguaje MOF (realmente

del modelo MOF del nivel M3), y la posibilidad de poder definir y consultar las correspondencias entre meta-datos en distintos niveles de abstracción de una manera efectiva. La herramienta busca proporcionar un interfaz intuitiva que facilite a los ingenieros manejar la complejidad que supone utilizar varios niveles abstracción diferentes. En resumen, METAMOD proporciona las siguientes funcionalidades:

- a) Creación, borrado, importación y exportación de modelos y metamodelos.
- b) Creación, modificación y borrado de los elementos de un modelo o metamodelo.

H.2.1. Formularios.

Desde el punto de vista de la interfaz de usuario, la herramienta consta de 5 formularios importantes:

- *Principal*: Contiene a todos los demás componentes visuales.
- *Administración de Metamodelos*: Es el encargado de presentar de forma jerárquica los distintos metamodelos y modelos que se encuentran en el repositorio de MANTIS.
- *Gestión de Metamodelos*: Permite la gestión de un metamodelo determinado presentando de forma jerárquica (en forma de árbol invertido) todos los elementos que contiene.
- *Gestión de Modelos*: Similar al anterior pero para los modelos, presenta jerárquicamente todos los elementos de un modelo.
- *Correspondencias*: Permite establecer y consultar las correspondencias entre elementos del modelo MOF (nivel M3) y sus instancias en un metamodelo (nivel M2) o entre los elementos de un metamodelo (nivel M2) y sus instancias en un modelo (nivel M1).

H.2.2. Funcionalidad.

A continuación se describen las principales funciones aportadas por la herramienta y la manera de usarlas. Para facilitar la comprensión, se han ordenado de igual manera a como deberían ser utilizadas en un caso real.

Crear Nuevo Metamodelo:

Para crear un nuevo metamodelo basta con pulsar sobre el menú “Archivo” y hacer clic sobre “Nuevo Metamodelo”. Aparece entonces el formulario de descripción de nuevos metamodelos (Figura H-1). Después de rellenar los datos pertinentes y pulsar “OK”, el metamodelo se añade a la lista de metamodelos presentada en el administrador de metamodelos. Inicialmente, el metamodelo únicamente contiene el paquete de alto nivel que contendrá los demás elementos del metamodelo.

Añadir Elementos a un Metamodelo:

Siguiendo el ejemplo, el siguiente paso es el de añadir las distintas clases MOF y asociaciones MOF al anterior metamodelo. Para ello, se debe modificar el paquete de alto nivel del metamodelo. Esto se realiza pulsando con el botón derecho del ratón en el formulario de Gestión de Metamodelos, sobre el paquete elegido y eligiendo la opción “Ver Contenidos” para abrir el formulario de modificación de dicho paquete. En la etiqueta “Contenidos” se muestran los distintos tipos de elementos que pueden añadirse a un paquete MOF: otros paquetes, asociaciones, clases y tipos de datos.

Descripcion Nuevo Metamodelo

Nombre: Metamodelo E/R

Version: 1 Nivel: 2

Documentacion

Propietario:

Contacto:

Descripcion Larga: Esta es la descripción del metamodelo E/R para la aplicación de MOF al campo de los datos

Descripción Corta: Metamodelo E/R

Exportador:

Version Exportador:

Id Exportador:

Nota:

OK Cancelar Aplicar Ayuda

Figura H-1. METAMOD: formulario para la descripción de nuevos metamodelos.

Clase

General Contenidos Supertipos

Nombre: Tipo Entidad

es_raiz ☐ es_hoja ☐

es_abstracto ☐ es_unico ☐

Anotacion:

Visibilidad: public_vis

OK Cancelar Aplicar Ayuda

Figura H-2. METAMOD: formulario para editar propiedades de las clases.

Para insertar una nueva clase MOF se elige la etiqueta “Contenidos” y, dentro de esta, la etiqueta “Clases”, pulsa el botón derecho de ratón y se elige la opción “Insertar”. Así, aparece el formulario “Clase”. En realidad, lo que se está haciendo es crear una nueva instancia de clase MOF, insertándola en la instancia de paquete MOF. Según establece el modelo MOF, para dicha instancia de clase MOF se establecen varias características (Figura H-2): permitir que cualquier elemento del modelo pueda usarla (visibilidad=public_vis) o no, ser abstracta o no (es_abstracto=falso), tener supertipos (es_raiz=falso) o no, ser supertipo (es_hoja=falso) o no, y poder existir varias instancias a la vez en el nivel M1 o no (es_unico=falso). De igual forma que se añaden clases MOF a un paquete MOF, se pueden añadir atributos a una clase MOF.

Una vez añadidas las diferentes clases MOF se deben insertar las asociaciones MOF. Esta tarea se realiza de forma similar a la anterior, activándose el formulario de “Asociaciones” para indicar sus características. Posteriormente a la definición de dichas características propias, se deben definir los dos “finales de asociación”.

Se continúan añadiendo todas las clases y asociaciones necesarias para construir el metamodelo correspondiente. En la Figura H-3 se muestra el formulario de “Administración de Metamodelos” con la lista de clases y asociaciones definidas para el metamodelo de ejemplo.

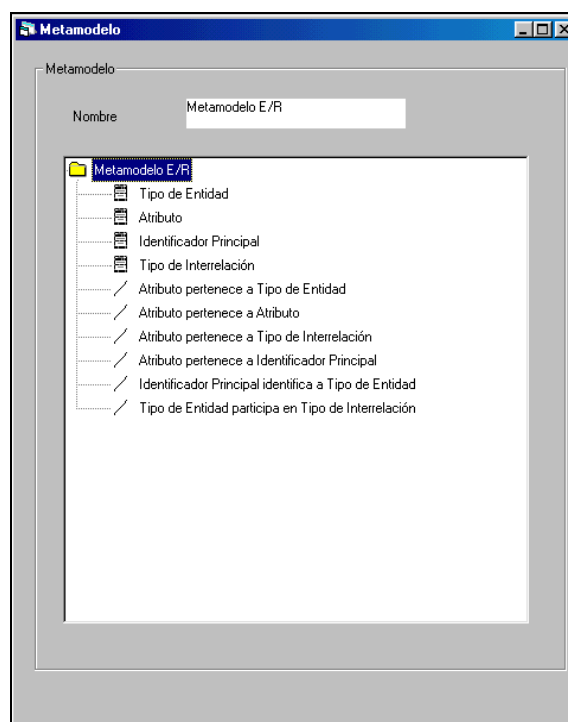


Figura H-3. METAMODO: formulario de administración de metamodelos.

Para guardar un metamodelo, modificado o recién creado, se activa dicho metamodelo en el formulario de “Administración de Metamodelos” y se elige la opción “Guardar” del menú “Archivo”.

Abrir o Importar Metamodelo:

En vez de tener que crear un metamodelo desde cero con METAMODO, es factible abrir un metamodelo diseñado previamente con METAMODO o con otra herramienta CASE (es decir,

importarlo). Esta segunda opción es especialmente recomendable en el caso de metamodelos (o modelos) complejos ya que será mucho más fácil realizar su diseño utilizando las facilidades gráficas proporcionadas habitualmente por las herramientas CASE. Para abrir o importar un metamodelo se debe elegir la opción “Abrir Metamodelo” del menú “Archivo”. El metamodelo debe estar almacenado en un documento en formato XMI y también debe estar disponible en otro archivo el DTD asociado. Se han realizado pruebas satisfactorias de importación/exportación de metamodelos y modelos con Rational Rose, es decir, abrir con METAMOD archivos XMI/DTD generados con Rational y abrir con Rational archivos XMI/DTD generados con METAMOD.

Crear Nuevo Modelo:

Una vez creados los metamodelos, lo normal será crear uno o varios modelos basados en dichos metamodelos. Para ello, en el formulario de “Administración de Metamodelos” se elige el metamodelo correspondiente y en el menú “Archivo” se elige la opción “Nuevo Modelo”. De esta forma se crea un nuevo modelo vacío que es instancia del metamodelo seleccionado. Este modelo se trata de igual forma que un metamodelo: se pueden modificar sus características y se le pueden añadir elementos de metamodelado de forma similar a cómo se hace con un metamodelo.

Añadir Elementos a un Modelo:

Se realiza de forma exactamente igual a como se añaden los elementos de un metamodelo. La única característica que los diferencia es que al añadir un objeto reflectivo se debe seleccionar la clase de nivel superior de la que será instancia. Por ejemplo, al añadir el objeto “edad” se debe indicar que es instancia de la clase “atributo”.

Abrir o Importar Modelo:

Es similar a la opción equivalente para los metamodelos (elegir la opción “Abrir Modelo” del menú “Archivo”) pero, previamente se debe haber elegido el metamodelo, del que el modelo es instancia, en el formulario “Administración de Metamodelos”.

Ver Correspondencias:

Como ya se ha comentado, la principal originalidad de METAMOD es la posibilidad de ver y editar las correspondencias entre elementos de distintos niveles de abstracción. Esta opción se activa eligiendo un metamodelo o modelo en el “Administrador de Metamodelos”, pulsando el botón derecho del ratón y eligiendo la opción “Ver Correspondencia Nivel Superior”. En el formulario “Correspondencias” aparecen dos ventanas. En la ventana izquierda se muestra el metamodelo de nivel superior al activado, es decir, aquel del cual el activado es instancia. Se puede navegar por su interior de forma similar a como se hace en el “Administrador de Metamodelos”. Para ver en la ventana derecha las instancias de nivel inferior asociadas con un elemento de dicho metamodelo se activa el elemento correspondiente en la ventana izquierda, se pulsa el botón derecho del ratón y se elige la opción “Ver Instancias Nivel Inferior”.

En la Figura H-4 se ha elegido el metamodelo de nivel M2 llamado “Metamodelo Básico de Procesos Software”, al elegir “Ver Correspondencias de Nivel Superior” se muestra a la derecha su meta-metamodelo del nivel M3, que siempre es el modelo MOF. En este último se ha elegido el elemento “Clase” (es decir, clase MOF) y al elegir la opción “Ver Instancias Nivel Inferior” se muestran las instancias de clase MOF, es decir, las clases del metamodelo M2: “Artefacto”, “Recurso”, “Actividad”, “Proyecto”, etc.

METAMOD es una aplicación multi-documento. Esto significa que es posible tener abiertos dos o más formularios “Correspondencias”; por ejemplo, uno el mostrado en la Figura

H-4 con las correspondencias M3-M2 y otro con las correspondencias M2-M1 para el mismo metamodelo M2 y un modelo M1 como, por ejemplo, el comentado en el anexo G. De esta forma se pueden comparar y analizar, de forma clara, las correspondencias entre los 3 niveles superiores de la arquitectura MOF.

Además, puesto que el modelo MOF es autodescriptivo (ver anexo C), es posible ver las clases MOF que contiene el propio modelo MOF.

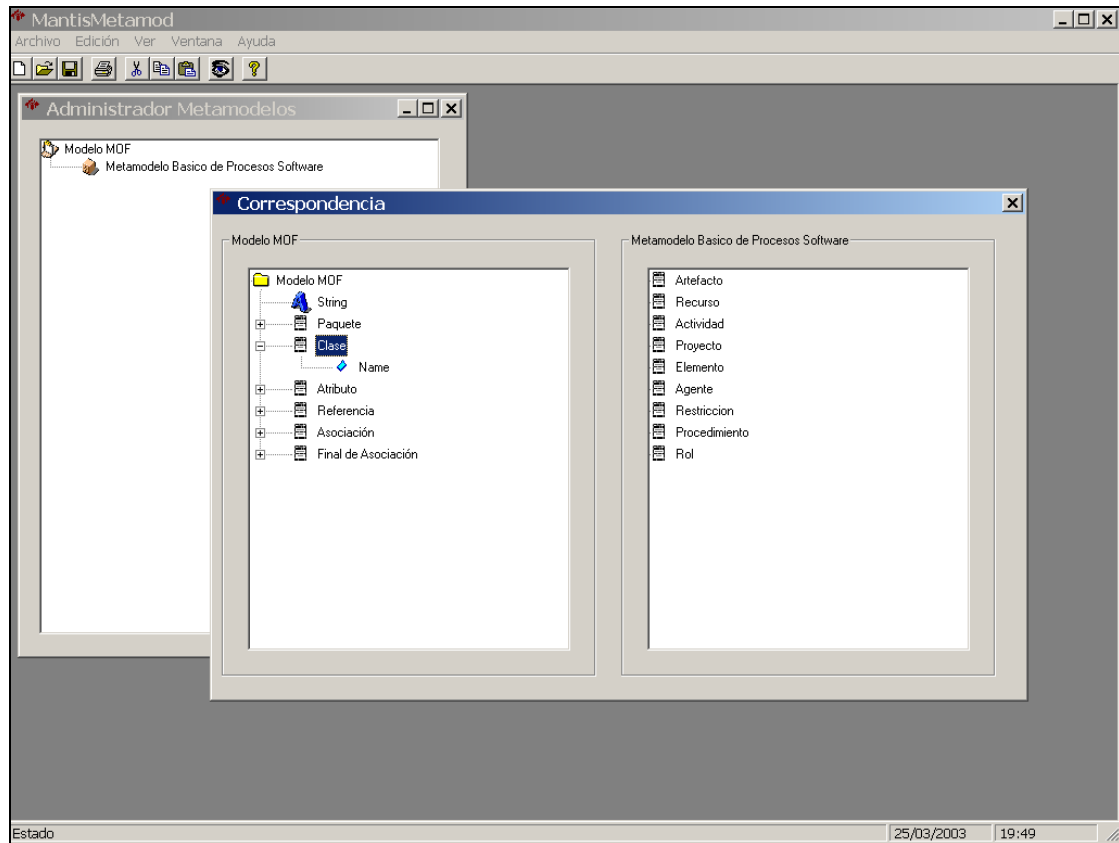


Figura H-4. METAMOD: Ver correspondencias entre elementos de diferentes niveles.

H.3. Manual de CREM.

CREM es una herramienta para ayudar a distribuir los recursos humanos entre las diversas peticiones de modificación (PM), pertenecientes a uno o varios proyectos, recibidas por una organización mantenedor. Está basada en un algoritmo de optimización de costes/beneficios (ver capítulo 6).

El formulario principal consta de los menús “Archivo”, “Informes” y “Ayuda”. También incluye una barra de herramientas con iconos para acceder a las funciones principales, así como una barra de estado, donde se informa del trabajo realizado en cada momento y de la cartera que está abierta.

H.3.1. Carteras.

Las funciones para gestionar las carteras de proyectos se utilizan de la siguiente manera:

Crear Nueva Cartera:

Se activa eligiendo la opción “Nueva Cartera” del menú “Archivo” o pulsando sobre el icono “Nueva” de la barra de herramientas. A continuación se abre un pequeño formulario para asignar nombre a la cartera. A partir de este momento el usuario podrá incorporar proyectos, recursos, etc., a dicha cartera.

Abrir/Eliminar/Cerrar Cartera:

Al elegir “Abrir Cartera” se muestra una lista desplegable con todas las carteras de proyectos almacenadas en el repositorio de la aplicación, para que el usuario elija una de ellas. Igual pasa para elegir la cartera que se desea eliminar. La opción “Cerrar Cartera” cierra la cartera actual. Sólo es posible tener abierta una cartera a la vez.

H.3.2. Proyectos.

En CREM cada cartera incluye uno o varios proyectos, los cuales a su vez reciben peticiones de modificación. Para atender dichas peticiones cada cartera de proyectos tiene asignados unos recursos. Las funciones para gestionar los proyectos son las siguientes:

Añadir Proyecto:

Una vez abierta una cartera o creada una nueva, para añadir un proyecto hay que situarse en la pestaña de proyectos y pulsar el botón secundario del ratón. En el menú contextual que aparece se deberá elegir la opción “Añadir”. Entonces aparece un formulario preguntando los datos del proyecto: nombre, ingresos y penalización. Una vez completados se añade dicho proyecto a la cartera activa pulsando el botón “Añadir”.

Modificar/Eliminar Proyecto:

También se pueden modificar los datos de un proyecto determinado. Para ello, basta con seleccionar primero el proyecto, pulsar el botón secundario del ratón y hacer clic sobre la opción “Modificar” del menú contextual. Aparecerá el formulario con los datos del proyecto. Para eliminar un proyecto se selecciona éste y se elige la opción “Eliminar” del menú de contexto correspondiente.

Figura H-5. CREM: Formulario para añadir peticiones de modificación.

H.3.3. Peticiones.

Añadir Petición:

Pulsando sobre la opción “Añadir Petición” del menú contextual se pueden añadir PM a la cartera activa. En el formulario que aparece (ver Figura H-5) se tienen que rellenar los campos “Identificador Proyecto” (seleccionando uno de los proyectos que hay en la cartera actual), el día de llegada de la PM (es posible hacerlo pulsando con el ratón sobre el calendario mostrado) y el número de horas que se han planificado para resolver esa PM.

Modificar/Eliminar Petición:

También es posible modificar los datos de una PM, o eliminar, con las opciones respectivas del menú de contexto.

H.3.4. Recursos.

Para realizar alguna operación con los recursos de una cartera de proyectos, lo primero que se debe hacer es elegir la pestaña de recursos y pulsar el botón secundario del ratón para elegir alguna de las siguientes opciones:

Añadir Recursos:

Se pueden adjudicar recursos a una cartera de dos formas: día a día, o para un intervalo de tiempo. Para lo primero hay que elegir en el menú la opción “Añadir (1 día)”. En el formulario que aparece se deben rellenar los siguientes datos: la fecha en la que se quiere asignar recursos (utilizar para ello el calendario situado a la derecha), las horas de recursos disponibles ese día, el coste por hora de recurso que pagará el cliente, y el coste por hora de recurso que asume el mantenedor.

Si se desean añadir recursos para un intervalo de tiempo hay que elegir en el menú contextual la opción “Añadir (intervalo)”. El formulario a rellenar en este caso es el que se muestra en la Figura H-6. Como se puede observar, se ofrece al usuario la posibilidad de indicar si los recursos están disponibles los días no laborables.

Añadir intervalo Recursos

Fecha Inicio: 02/12/01

Días: 20

Horas Disponibles: 24

Coste por Hora (Cliente): 5000

Coste por Hora (Organiz): 2300

☒ Trabajar días no laborables

Añadir

Calendar: diciembre 2001

lun	mar	mié	jue	vie	sáb	dom
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Hoy: 04/12/01

Figura H-6. CREM: Formulario para añadir recursos para un intervalo de tiempo.

Modificar/Eliminar Recursos:

En lo referente a las opciones de modificar o eliminar recursos funcionan igual que las equivalentes de proyectos.

H.3.5. Informes.Generar e Imprimir Informe:

Con CREM se pueden obtener informes de proyectos, peticiones, recursos y de las asignaciones de recursos a cada una de las peticiones cada día. Estos informes se pueden conseguir de tres formas alternativas: pulsando sobre el icono correspondiente de la barra de herramientas; eligiendo el menú “Informes” y pinchando en el nombre del informe que se desea obtener; o usando los menús de contexto de proyectos, peticiones o recursos y eligiendo la opción “Informes”. En cualquiera de estos casos, aparece un formulario sencillo para elegir el campo por el que se desea ordenar y el criterio de ordenación (ascendente o descendente). Una vez elegidas ambas opciones se visualizará el informe en pantalla con la opción de enviar a impresora pulsando el icono correspondiente. La imagen en pantalla del informe obtenido se puede ampliar o reducir con el icono de “zoom”. También se puede navegar por las diversas páginas con los botones de “avance” y “retroceso”.

Exportar Informe:

Otra opción del generador de informes de CREM es la posibilidad de exportar el informe a los formatos HTML o texto plano (ASCII, Unicode o UTF-8).

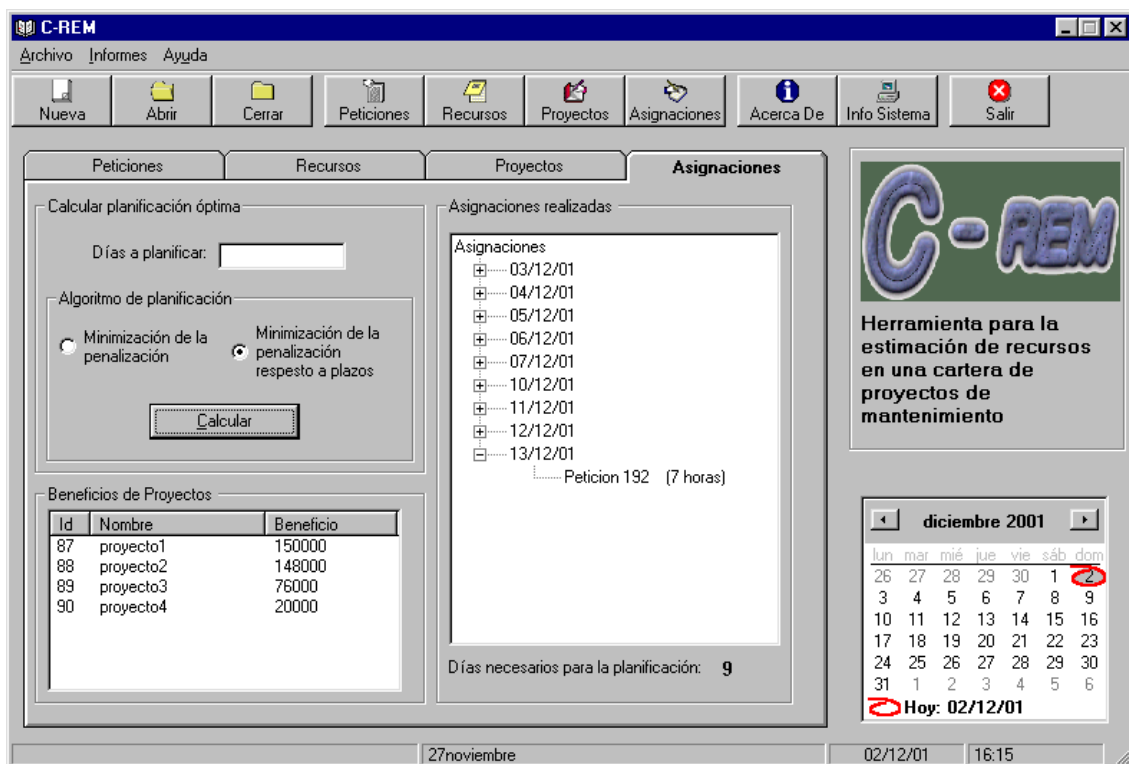


Figura H-7. CREM: Asignación de recursos a peticiones de modificación.

H.3.6. Asignaciones.

Una vez introducidos todos los proyectos, peticiones y recursos necesarios, se pueden calcular las asignaciones óptimas de recursos por día y PM (Figura H-7). Existen dos algoritmos para realizar este cálculo (ver detalles en capítulo 6): minimización de la penalización, y Minimización de la penalización respecto a los plazos de entrega. El primero atiende primero las PM que pertenecen a proyectos que cuentan con mayor penalización, mientras que el segundo tiene en cuenta también la penalización que tiene el proyecto al que pertenece la PM y los días que le quedan para comenzar a penalizar por dicha PM.

Además de elegir el tipo de cálculo, el usuario debe indicar el número de días que se quieren calcular. Si el número de días es inferior a los días necesarios empleando todos los recursos que tenga asignados esa cartera, se indicará con un mensaje cual es el número mínimo de días que se necesitan. Si no hay asignados recursos suficientes para poder realizar todas las PM, también se indicará con un mensaje para que se aumenten los recursos disponibles en la cartera.

Una vez el número de días a calcular válido, se puede pulsar el botón “Calcular”. Los resultados se muestran en dos listas. La primera, en forma de lista desplegable (Figura H-8), muestra en el centro de la pantalla las asignaciones de recursos humanos (en horas-persona). La segunda, con forma de tabla situada en la parte inferior izquierda de la pantalla, permite observar los beneficios que consigue cada uno de los proyectos de la cartera con la distribución de recursos propuesta.

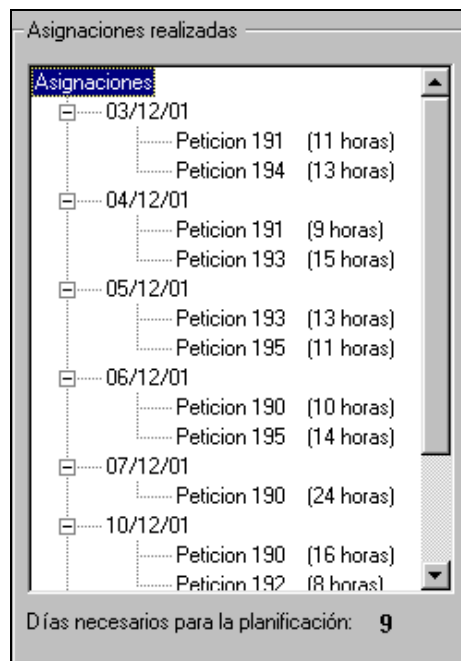


Figura H-8. CREM: Desglose de asignaciones de recursos por días y petición de modificación.

I. Ejemplos de Documentos DTD y XMI.

En este anexo se presentan varios ejemplos de documentos de tipo DTD y XMI generados automáticamente por las herramientas RepManager y/o METAMOD incluidas en el Entorno MANTIS. En concreto, los documentos incluidos son los generados a partir del metamodelo y del modelo indicados en el anexo G.

I.1. Documentos para Correspondencias M3-M2.

El paquete básico adaptado para el metamodelo genérico de proceso software (ver anexo G) es representado en el repositorio de MANTIS mediante un documento XMI, que se presenta a continuación, cuyo DTD es el del modelo MOF, que es único y universal³⁰.

I.1.1. Documento XMI del Nivel M2.

Este documento ha sido generado con la herramienta METAMOD que, a su vez, ha invocado el servicio “GenerarDocumentoXMI” de RepManager utilizando el valor 2 para la propiedad ‘nivel’ del parámetro ‘Modelo o metamodelo’. En el anexo H se puede consultar el manual del programador para invocar este servicio.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE XMI (View Source for full doctype...)>
<XMI xmi.version="1.0">
  <XMI.header>
    <XMI.documentation>
      Documentacion asociada:
      <XMI.exporter>Mantis-Metamod</XMI.exporter>
      <XMI.exporterVersion>1.0</XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="Metamodelo Basico de Procesos Software" xmi.version="1.0" />
    <XMI.metamodel xmi.name="MOF" xmi.version="1.3" />
  </XMI.header>
  <XMI.content>
    <!--
    METAMODELO: org.omg.mof.Model
    -->
    <!--
    PAQUETE: Basico
    -->
    <Model.Package xmi.id="a3A23FAFC03CA">
      <Model.ModelElement.name>Basico</Model.ModelElement.name>
      <Model.ModelElement.annotation />
      <Model.GeneralizableElement.isRoot xmi.value="false" />
      <Model.GeneralizableElement.isLeaf xmi.value="false" />
      <Model.GeneralizableElement.isAbstract xmi.value="false" />
      <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    </Model.Namespace.contents>
  </XMI.content>
</XMI>
```

³⁰ El archivo DTD del nivel M3 correspondiente al modelo MOF está incluido en el CD que acompaña a esta tesis.

CLASE: Artefacto

```
-->
<Model.Class xmi.id="a3A241DFC001E">
  <Model.ModelElement.name>Artefacto</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.GeneralizableElement.isRoot xmi.value="false" />
  <Model.GeneralizableElement.isLeaf xmi.value="false" />
  <Model.GeneralizableElement.isAbstract xmi.value="false" />
  <Model.GeneralizableElement.visibility xmi.value="public_vis" />
  <Model.Class.isSingleton xmi.value="false" />
  <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
  </Model.ModelElement.container>
  <Model.GeneralizableElement.supertypes>
    <Model.GeneralizableElement xmi.idref="a3D74DD9A0122" />
  </Model.GeneralizableElement.supertypes>
  <Model.Namespace.contents>
    - <!--
      REFERENCIA: referencia_a_compuesto
    -->
    <Model.Reference
      xmi.id="a7FB01F40F01011D6B74F000D05967378">

        <Model.ModelElement.name>referencia_a_compuesto</
          Model.ModelElement.name>
        <Model.ModelElement.annotation />
        <Model.Feature.scope xmi.value="instance_level" />
        <Model.Feature.visibility xmi.value="public_vis" />
        <Model.StructuralFeature.multiplicity>
          <XMI.field>0</XMI.field>
          <XMI.field>-1</XMI.field>
          <XMI.field>>false</XMI.field>
          <XMI.field>>true</XMI.field>
        </Model.StructuralFeature.multiplicity>
        <Model.StructuralFeature.isChangeable xmi.value="true" />
        <Model.ModelElement.container>
          <Model.Namespace xmi.idref="a3A241DFC001E" />
        </Model.ModelElement.container>
        <Model.TypedElement.type>
          <Model.Classifier xmi.idref="a3A241DFC001E" />
        </Model.TypedElement.type>
        <Model.Reference.referencedEnd>
          <Model.AssociationEnd xmi.idref="a3D85B31002DE" />
        </Model.Reference.referencedEnd>
      </Model.Reference>
    - <!--
      REFERENCIA: referencia_a_actividad_de_entrada
    -->
    <Model.Reference
      xmi.id="a7FB01F64F01011D6B74F000D05967378">

        <Model.ModelElement.name>referencia_a_actividad_de
          _entrada</Model.ModelElement.name>
        <Model.ModelElement.annotation />
        <Model.Feature.scope xmi.value="instance_level" />
        <Model.Feature.visibility xmi.value="public_vis" />
        <Model.StructuralFeature.multiplicity>
          <XMI.field>1</XMI.field>
          <XMI.field>-1</XMI.field>
          <XMI.field>>false</XMI.field>
          <XMI.field>>true</XMI.field>
        </Model.StructuralFeature.multiplicity>
        <Model.StructuralFeature.isChangeable xmi.value="true" />
        <Model.ModelElement.container>
          <Model.Namespace xmi.idref="a3A241DFC001E" />
        </Model.ModelElement.container>
        <Model.TypedElement.type>
          <Model.Classifier xmi.idref="a3A241DFC0024" />
        </Model.TypedElement.type>
        <Model.Reference.referencedEnd>
          <Model.AssociationEnd xmi.idref="a3D85C68E0168" />
        </Model.Reference.referencedEnd>
      </Model.Reference>
    - <!--
  </Model.Namespace.contents>
</Model.Class>
```

```

        </Model.Reference.referencedEnd>
    </Model.Reference>
- <!--
REFERENCIA: referencia_a_actividad_de_salida
-->
=
        <Model.Reference
            xmi.id="a7FB01F68F01011D6B74F000D05967378">

            <Model.ModelElement.name>referencia_a_actividad_de
            _salida</Model.ModelElement.name>
            <Model.ModelElement.annotation />
            <Model.Feature.scope xmi.value="instance_level" />
            <Model.Feature.visibility xmi.value="public_vis" />
            = <Model.StructuralFeature.multiplicity>
                <XMI.field>1</XMI.field>
                <XMI.field>-1</XMI.field>
                <XMI.field>>false</XMI.field>
                <XMI.field>true</XMI.field>
            </Model.StructuralFeature.multiplicity>
            <Model.StructuralFeature.isChangeable xmi.value="true" />
            = <Model.ModelElement.container>
                <Model.Namespace xmi.idref="a3A241DFC001E" />
            </Model.ModelElement.container>
            = <Model.TypedElement.type>
                <Model.Classifier xmi.idref="a3A241DFC0024" />
            </Model.TypedElement.type>
            = <Model.Reference.referencedEnd>
                <Model.AssociationEnd xmi.idref="a3D85C69403BF" />
            </Model.Reference.referencedEnd>
        </Model.Reference>
    </Model.Namespace.contents>
</Model.Class>
- <!--
CLASE: Recurso
-->
= <Model.Class xmi.id="a3A241DFC0029">
    <Model.ModelElement.name>Recurso</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="false" />
    <Model.GeneralizableElement.isLeaf xmi.value="false" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Class.isSingleton xmi.value="false" />
    = <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    </Model.ModelElement.container>
    = <Model.GeneralizableElement.supertypes>
        <Model.GeneralizableElement xmi.idref="a3D74DD9A0122" />
    </Model.GeneralizableElement.supertypes>
    <Model.Namespace.contents />
</Model.Class>
- <!--
CLASE: Actividad
-->
= <Model.Class xmi.id="a3A241DFC0024">
    <Model.ModelElement.name>Actividad</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="false" />
    <Model.GeneralizableElement.isLeaf xmi.value="false" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Class.isSingleton xmi.value="false" />
    = <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    </Model.ModelElement.container>
    = <Model.GeneralizableElement.supertypes>
        <Model.GeneralizableElement xmi.idref="a3D74DD9A0122" />
    </Model.GeneralizableElement.supertypes>
    = <Model.Namespace.contents>
        - <!--
        ATRIBUTO: nivelabstraccion

```

```

-->
<Model.Attribute xmi.id="a3D74704301A1">

    <Model.ModelElement.name>nivelabstraccion</Model.M
odelElement.name>
    <Model.ModelElement.annotation />
    <Model.Feature.scope xmi.value="instance_level" />
    <Model.Feature.visibility xmi.value="private_vis" />
    <Model.StructuralFeature.multiplicity>
        <XMI.field>1</XMI.field>
        <XMI.field>1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>false</XMI.field>
    </Model.StructuralFeature.multiplicity>
    <Model.StructuralFeature.isChangeable xmi.value="true" />
    <Model.Attribute.isDerived xmi.value="false" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A241DFC0024" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
        <Model.Classifier xmi.idref="G.49" />
    </Model.TypedElement.type>
</Model.Attribute>
- <!--
REFERENCIA: referencia_a_procedimiento_utilizado
-->

<Model.Reference
xmi.id="a7FB01F44F01011D6B74F000D05967378">

    <Model.ModelElement.name>referencia_a_procedimient
o_utilizado</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.Feature.scope xmi.value="instance_level" />
    <Model.Feature.visibility xmi.value="public_vis" />
    <Model.StructuralFeature.multiplicity>
        <XMI.field>1</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
    </Model.StructuralFeature.multiplicity>
    <Model.StructuralFeature.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A241DFC0024" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D85C7DF024C" />
    </Model.TypedElement.type>
    <Model.Reference.referencedEnd>
        <Model.AssociationEnd xmi.idref="a3D85C809013C" />
    </Model.Reference.referencedEnd>
</Model.Reference>
- <!--
REFERENCIA: referencia_a_procedimiento
-->

<Model.Reference
xmi.id="a7FB01F48F01011D6B74F000D05967378">

    <Model.ModelElement.name>referencia_a_procedimient
o</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.Feature.scope xmi.value="instance_level" />
    <Model.Feature.visibility xmi.value="public_vis" />
    <Model.StructuralFeature.multiplicity>
        <XMI.field>1</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
    </Model.StructuralFeature.multiplicity>
    <Model.StructuralFeature.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A241DFC0024" />

```

```

</Model.ModelElement.container>
<Model.TypedElement.type>
  <Model.Classifier xmi.idref="a3D85C7DF024C" />
</Model.TypedElement.type>
<Model.Reference.referencedEnd>
  <Model.AssociationEnd xmi.idref="a3D85C80F0086" />
</Model.Reference.referencedEnd>
</Model.Reference>
- <!--
REFERENCIA: referencia_a_recurso
-->
<Model.Reference
  xmi.id="a7FB01F50F01011D6B74F000D05967378">

  <Model.ModelElement.name>referencia_a_recurso</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.Feature.scope xmi.value="instance_level" />
  <Model.Feature.visibility xmi.value="public_vis" />
  <Model.StructuralFeature.multiplicity>
    <XMI.field>0</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
  </Model.StructuralFeature.multiplicity>
  <Model.StructuralFeature.isChangeable xmi.value="true" />
  <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A241DFC0024" />
  </Model.ModelElement.container>
  <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3A241DFC0029" />
  </Model.TypedElement.type>
  <Model.Reference.referencedEnd>
    <Model.AssociationEnd xmi.idref="a3D85C92D03CC" />
  </Model.Reference.referencedEnd>
</Model.Reference>
</Model.Namespace.contents>
</Model.Class>
- <!--
CLASE: Proyecto
-->
<Model.Class xmi.id="a3D74DD4F019D">
  <Model.ModelElement.name>Proyecto</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.GeneralizableElement.isRoot xmi.value="false" />
  <Model.GeneralizableElement.isLeaf xmi.value="false" />
  <Model.GeneralizableElement.isAbstract xmi.value="false" />
  <Model.GeneralizableElement.visibility xmi.value="public_vis" />
  <Model.Class.isSingleton xmi.value="false" />
  <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
  </Model.ModelElement.container>
  <Model.Namespace.contents>
    - <!--
    REFERENCIA: referencia_a_elemento
    -->
    <Model.Reference
      xmi.id="a7FB01F30F01011D6B74F000D05967378">

      <Model.ModelElement.name>referencia_a_elemento</Model.ModelElement.name>
      <Model.ModelElement.annotation />
      <Model.Feature.scope xmi.value="instance_level" />
      <Model.Feature.visibility xmi.value="public_vis" />
      <Model.StructuralFeature.multiplicity>
        <XMI.field>1</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
      </Model.StructuralFeature.multiplicity>
    </Model.Reference>
  </Model.Namespace.contents>
</Model.Class>

```

```

        <Model.StructuralFeature.isChangeable xmi.value="true" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D74DD4F019D" />
    - <Model.ModelElement.container>
    - <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D74DD9A0122" />
    - </Model.TypedElement.type>
    - <Model.Reference.referencedEnd>
        <Model.AssociationEnd xmi.idref="a3D74DE320019" />
    - </Model.Reference.referencedEnd>
    - </Model.Reference>
- <!--
REFERENCIA: referencia_a_sub-proyecto
-->
=
    <Model.Reference
        xmi.id="a7FB01F34F01011D6B74F000D05967378">
        <Model.ModelElement.name>referencia_a_sub-
            proyecto</Model.ModelElement.name>
        <Model.ModelElement.annotation />
        <Model.Feature.scope xmi.value="instance_level" />
        <Model.Feature.visibility xmi.value="public_vis" />
    - <Model.StructuralFeature.multiplicity>
        <XMI.field>0</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>true</XMI.field>
    - </Model.StructuralFeature.multiplicity>
        <Model.StructuralFeature.isChangeable xmi.value="true" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D74DD4F019D" />
    - </Model.ModelElement.container>
    - <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D74DD4F019D" />
    - </Model.TypedElement.type>
    - <Model.Reference.referencedEnd>
        <Model.AssociationEnd xmi.idref="a3D74DE4C0192" />
    - </Model.Reference.referencedEnd>
    - </Model.Reference>
- <!--
REFERENCIA: referencia_a_restriccion
-->
=
    <Model.Reference
        xmi.id="a7FB01F3CF01011D6B74F000D05967378">
        <Model.ModelElement.name>referencia_a_restriccion</
            Model.ModelElement.name>
        <Model.ModelElement.annotation />
        <Model.Feature.scope xmi.value="instance_level" />
        <Model.Feature.visibility xmi.value="public_vis" />
    - <Model.StructuralFeature.multiplicity>
        <XMI.field>1</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>true</XMI.field>
    - </Model.StructuralFeature.multiplicity>
        <Model.StructuralFeature.isChangeable xmi.value="true" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D74DD4F019D" />
    - </Model.ModelElement.container>
    - <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D74DE1201FE" />
    - </Model.TypedElement.type>
    - <Model.Reference.referencedEnd>
        <Model.AssociationEnd xmi.idref="a3D85B2A2037C" />
    - </Model.Reference.referencedEnd>
    - </Model.Reference>
    - </Model.Namespace.contents>
    - </Model.Class>
- <!--
CLASE: Elemento
-->

```

```

<Model.Class xmi.id="a3D74DD9A0122">
  <Model.ModelElement.name>Elemento</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.GeneralizableElement.isRoot xmi.value="false" />
  <Model.GeneralizableElement.isLeaf xmi.value="false" />
  <Model.GeneralizableElement.isAbstract xmi.value="false" />
  <Model.GeneralizableElement.visibility xmi.value="public_vis" />
  <Model.Class.isSingleton xmi.value="false" />
  <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
  </Model.ModelElement.container>
  <Model.Namespace.contents>
    <!--
    ATRIBUTO: nombre
    -->
    <Model.Attribute xmi.id="a3DC68E280366">
      <Model.ModelElement.name>nombre</Model.ModelElement.name>
      <Model.ModelElement.annotation />
      <Model.Feature.scope xmi.value="instance_level" />
      <Model.Feature.visibility xmi.value="private_vis" />
      <Model.StructuralFeature.multiplicity>
        <XMI.field>1</XMI.field>
        <XMI.field>1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>false</XMI.field>
      </Model.StructuralFeature.multiplicity>
      <Model.StructuralFeature.isChangeable xmi.value="true" />
      <Model.Attribute.isDerived xmi.value="false" />
      <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D74DD9A0122" />
      </Model.ModelElement.container>
      <Model.TypedElement.type>
        <Model.Classifier xmi.idref="G.49" />
      </Model.TypedElement.type>
      </Model.Attribute>
    </Model.Namespace.contents>
  </Model.Class>
  <!--
  CLASE: Agente
  -->
  <Model.Class xmi.id="a3D74DE040262">
    <Model.ModelElement.name>Agente</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="false" />
    <Model.GeneralizableElement.isLeaf xmi.value="false" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Class.isSingleton xmi.value="false" />
    <Model.ModelElement.container>
      <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    </Model.ModelElement.container>
    <Model.GeneralizableElement.supertypes>
      <Model.GeneralizableElement xmi.idref="a3D74DD9A0122" />
    </Model.GeneralizableElement.supertypes>
    <Model.Namespace.contents>
      <!--
      REFERENCIA: referencia_a_recurso
      -->
      <Model.Reference
        xmi.id="a7FB01F54F01011D6B74F000D05967378">
        <Model.ModelElement.name>referencia_a_recurso</Model.ModelElement.name>
        <Model.ModelElement.annotation />
        <Model.Feature.scope xmi.value="instance_level" />
        <Model.Feature.visibility xmi.value="public_vis" />
        <Model.StructuralFeature.multiplicity>
          <XMI.field>0</XMI.field>
          <XMI.field>-1</XMI.field>

```

```

        <XML.field>false</XML.field>
        <XML.field>true</XML.field>
    </Model.StructuralFeature.multiplicity>
    <Model.StructuralFeature.isChangeable xmi.value="true" />
<Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D74DE040262" />
</Model.ModelElement.container>
<Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3A241DFC0029" />
</Model.TypedElement.type>
<Model.Reference.referencedEnd>
    <Model.AssociationEnd xmi.idref="a3D85C9BA0197" />
</Model.Reference.referencedEnd>
</Model.Reference>
- <!--
REFERENCIA: referencia_a_rol
-->
=
        <Model.Reference
            xmi.id="a7FB01F5CF01011D6B74F000D05967378">

            <Model.ModelElement.name>referencia_a_rol</Model.M
            odelElement.name>
            <Model.ModelElement.annotation />
            <Model.Feature.scope xmi.value="instance_level" />
            <Model.Feature.visibility xmi.value="public_vis" />
        <Model.StructuralFeature.multiplicity>
            <XML.field>0</XML.field>
            <XML.field>-1</XML.field>
            <XML.field>false</XML.field>
            <XML.field>true</XML.field>
        </Model.StructuralFeature.multiplicity>
        <Model.StructuralFeature.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D74DE040262" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D85C9F200AF" />
    </Model.TypedElement.type>
    <Model.Reference.referencedEnd>
        <Model.AssociationEnd xmi.idref="a3D85CA7600BC"
        />
    </Model.Reference.referencedEnd>
</Model.Reference>
- <!--
REFERENCIA: referencia_a_actividad
-->
=
        <Model.Reference
            xmi.id="a7FB01F60F01011D6B74F000D05967378">

            <Model.ModelElement.name>referencia_a_actividad</M
            odel.ModelElement.name>
            <Model.ModelElement.annotation />
            <Model.Feature.scope xmi.value="instance_level" />
            <Model.Feature.visibility xmi.value="public_vis" />
        <Model.StructuralFeature.multiplicity>
            <XML.field>0</XML.field>
            <XML.field>-1</XML.field>
            <XML.field>false</XML.field>
            <XML.field>true</XML.field>
        </Model.StructuralFeature.multiplicity>
        <Model.StructuralFeature.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D74DE040262" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3A241DFC0024" />
    </Model.TypedElement.type>
    <Model.Reference.referencedEnd>
        <Model.AssociationEnd xmi.idref="a3D85CABB028E"
        />
    </Model.Reference.referencedEnd>
</Model.Reference>

```



```

    </Model.Namespace.contents>
  </Model.Class>
- <!--
CLASE: Restriccion
-->
- <Model.Class xmi.id="a3D74DE1201FE">
  <Model.ModelElement.name>Restriccion</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.GeneralizableElement.isRoot xmi.value="false" />
  <Model.GeneralizableElement.isLeaf xmi.value="false" />
  <Model.GeneralizableElement.isAbstract xmi.value="false" />
  <Model.GeneralizableElement.visibility xmi.value="public_vis" />
  <Model.Class.isSingleton xmi.value="false" />
  <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
  </Model.ModelElement.container>
- <Model.Namespace.contents>
  - <!--
  REFERENCIA: referencia_a_elemento
  -->
  =
    <Model.Reference
      xmi.id="a7FB01F38F01011D6B74F000D05967378">

      <Model.ModelElement.name>referencia_a_elemento</Model.ModelElement.name>
      <Model.ModelElement.annotation />
      <Model.Feature.scope xmi.value="instance_level" />
      <Model.Feature.visibility xmi.value="public_vis" />
      <Model.StructuralFeature.multiplicity>
        <XMI.field>1</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
      </Model.StructuralFeature.multiplicity>
      <Model.StructuralFeature.isChangeable xmi.value="true" />
      <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D74DE1201FE" />
      </Model.ModelElement.container>
      <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D74DD9A0122" />
      </Model.TypedElement.type>
      <Model.Reference.referencedEnd>
        <Model.AssociationEnd xmi.idref="a3D74DF8A0310" />
      </Model.Reference.referencedEnd>
    </Model.Reference>
  </Model.Namespace.contents>
</Model.Class>
- <!--
CLASE: Procedimiento
-->
- <Model.Class xmi.id="a3D85C7DF024C">
  <Model.ModelElement.name>Procedimiento</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.GeneralizableElement.isRoot xmi.value="false" />
  <Model.GeneralizableElement.isLeaf xmi.value="false" />
  <Model.GeneralizableElement.isAbstract xmi.value="false" />
  <Model.GeneralizableElement.visibility xmi.value="public_vis" />
  <Model.Class.isSingleton xmi.value="false" />
  <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
  </Model.ModelElement.container>
- <Model.Namespace.contents>
  - <!--
  REFERENCIA: referencia_a_artefacto
  -->
  =
    <Model.Reference
      xmi.id="a7FB01F4CF01011D6B74F000D05967378">

      <Model.ModelElement.name>referencia_a_artefacto</Model.ModelElement.name>
      <Model.ModelElement.annotation />

```

```

        <Model.Feature.scope xmi.value="instance_level" />
        <Model.Feature.visibility xmi.value="public_vis" />
    - <Model.StructuralFeature.multiplicity>
        <XMI.field>0</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
    </Model.StructuralFeature.multiplicity>
    <Model.StructuralFeature.isChangeable xmi.value="true" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D85C7DF024C" />
    </Model.ModelElement.container>
    - <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3A241DFC001E" />
    </Model.TypedElement.type>
    - <Model.Reference.referencedEnd>
        <Model.AssociationEnd xmi.idref="a3D85C86C0111" />
    </Model.Reference.referencedEnd>
    </Model.Reference>
    </Model.Namespace.contents>
</Model.Class>
- <!--
CLASE: Rol
-->
- <Model.Class xmi.id="a3D85C9F200AF">
    <Model.ModelElement.name>Rol</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="false" />
    <Model.GeneralizableElement.isLeaf xmi.value="false" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Class.isSingleton xmi.value="false" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    </Model.ModelElement.container>
    - <Model.Namespace.contents>
        - <!--
        REFERENCIA: referencia_a_actividad
        -->
        - <Model.Reference
            xmi.id="a7FB01F58F01011D6B74F000D05967378">

                <Model.ModelElement.name>referencia_a_actividad</M
                odel.ModelElement.name>
            <Model.ModelElement.annotation />
            <Model.Feature.scope xmi.value="instance_level" />
            <Model.Feature.visibility xmi.value="public_vis" />
            - <Model.StructuralFeature.multiplicity>
                <XMI.field>1</XMI.field>
                <XMI.field>-1</XMI.field>
                <XMI.field>>false</XMI.field>
                <XMI.field>>true</XMI.field>
            </Model.StructuralFeature.multiplicity>
            <Model.StructuralFeature.isChangeable xmi.value="true" />
            - <Model.ModelElement.container>
                <Model.Namespace xmi.idref="a3D85C9F200AF" />
            </Model.ModelElement.container>
            - <Model.TypedElement.type>
                <Model.Classifier xmi.idref="a3A241DFC0024" />
            </Model.TypedElement.type>
            - <Model.Reference.referencedEnd>
                <Model.AssociationEnd xmi.idref="a3D85CA2D0060" />
            </Model.Reference.referencedEnd>
            </Model.Reference>
        </Model.Namespace.contents>
    </Model.Class>
    - <!--
    ASOCIACION: componer
    -->
    - <Model.Association xmi.id="a3D73417D02C4">
        <Model.ModelElement.name>componer</Model.ModelElement.name>

```

```

<Model.ModelElement.annotation />
<Model.GeneralizableElement.isRoot xmi.value="true" />
<Model.GeneralizableElement.isLeaf xmi.value="true" />
<Model.GeneralizableElement.isAbstract xmi.value="false" />
<Model.GeneralizableElement.visibility xmi.value="public_vis" />
<Model.Association.isDerived xmi.value="false" />
<Model.ModelElement.container>
  <Model.Namespace xmi.idref="a3A23FAFC03CA" />
</Model.ModelElement.container>
<Model.Namespace.contents>
  - <!--
    FINAL DE ASOCIACIÓN: super-actividad
    -->
  <Model.AssociationEnd xmi.id="a3D7341820235">
    <Model.ModelElement.name>super-
      actividad</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="shared" />
    <Model.AssociationEnd.multiplicity>
      <XMI.field>0</XMI.field>
      <XMI.field>-1</XMI.field>
      <XMI.field>>false</XMI.field>
      <XMI.field>>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
      <Model.Namespace xmi.idref="a3D73417D02C4" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
      <Model.Classifier xmi.idref="a3A241DFC0024" />
    </Model.TypedElement.type>
    </Model.AssociationEnd>
  - <!--
    FINAL DE ASOCIACIÓN: sub-actividad
    -->
  <Model.AssociationEnd xmi.id="a3D7341820249">
    <Model.ModelElement.name>sub-
      actividad</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
    <Model.AssociationEnd.multiplicity>
      <XMI.field>0</XMI.field>
      <XMI.field>-1</XMI.field>
      <XMI.field>>false</XMI.field>
      <XMI.field>>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
      <Model.Namespace xmi.idref="a3D73417D02C4" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
      <Model.Classifier xmi.idref="a3A241DFC0024" />
    </Model.TypedElement.type>
    </Model.AssociationEnd>
  </Model.Namespace.contents>
</Model.Association>
- <!--
ASOCIACION: contiene
-->
<Model.Association xmi.id="a3D74DE3001ED">
  <Model.ModelElement.name>contiene</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.GeneralizableElement.isRoot xmi.value="true" />
  <Model.GeneralizableElement.isLeaf xmi.value="true" />
  <Model.GeneralizableElement.isAbstract xmi.value="false" />
  <Model.GeneralizableElement.visibility xmi.value="public_vis" />
  <Model.Association.isDerived xmi.value="false" />
  <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
  </Model.ModelElement.container>

```

```

</Model.ModelElement.container>
- <!--
  FINAL DE ASOCIACIÓN: elemento
-->
- <Model.AssociationEnd xmi.id="a3D74DE320019">
  <Model.ModelElement.name>elemento</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.AssociationEnd.isNavigable xmi.value="true" />
  <Model.AssociationEnd.aggregation xmi.value="none" />
  - <Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
  </Model.AssociationEnd.multiplicity>
  <Model.AssociationEnd.isChangeable xmi.value="true" />
  - <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D74DE3001ED" />
  </Model.ModelElement.container>
  - <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3D74DD9A0122" />
  </Model.TypedElement.type>
</Model.AssociationEnd>
- <!--
  FINAL DE ASOCIACIÓN: proyecto
-->
- <Model.AssociationEnd xmi.id="a3D74DE320023">
  <Model.ModelElement.name>proyecto</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.AssociationEnd.isNavigable xmi.value="false" />
  <Model.AssociationEnd.aggregation xmi.value="shared" />
  - <Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
  </Model.AssociationEnd.multiplicity>
  <Model.AssociationEnd.isChangeable xmi.value="true" />
  - <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D74DE3001ED" />
  </Model.ModelElement.container>
  - <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3D74DD4F019D" />
  </Model.TypedElement.type>
</Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
  ASOCIACION: contiene
-->
- <Model.Association xmi.id="a3D74DE400023">
  <Model.ModelElement.name>contiene</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.GeneralizableElement.isRoot xmi.value="true" />
  <Model.GeneralizableElement.isLeaf xmi.value="true" />
  <Model.GeneralizableElement.isAbstract xmi.value="false" />
  <Model.GeneralizableElement.visibility xmi.value="public_vis" />
  <Model.Association.isDerived xmi.value="false" />
  - <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
  </Model.ModelElement.container>
  - <Model.Namespace.contents>
    - <!--
      FINAL DE ASOCIACIÓN: sub-proyecto
    -->
    - <Model.AssociationEnd xmi.id="a3D74DE4C0192">

```

```

        <Model.ModelElement.name>sub-
    proyecto</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="true" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
    <Model.AssociationEnd.multiplicity>
    <XMI.field>0</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D74DE400023" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3D74DD4F019D" />
    </Model.TypedElement.type>
    </Model.AssociationEnd>
    - <!--
    FINAL DE ASOCIACIÓN: proyecto
    -->
    <Model.AssociationEnd xmi.id="a3D74DE4C019C">

        <Model.ModelElement.name>proyecto</Model.ModelEle
        ment.name>
        <Model.ModelElement.annotation />
        <Model.AssociationEnd.isNavigable xmi.value="false" />
        <Model.AssociationEnd.aggregation xmi.value="shared" />
        <Model.AssociationEnd.multiplicity>
        <XMI.field>0</XMI.field>
        <XMI.field>1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>false</XMI.field>
        </Model.AssociationEnd.multiplicity>
        <Model.AssociationEnd.isChangeable xmi.value="true" />
        <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D74DE400023" />
        </Model.ModelElement.container>
        <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D74DD4F019D" />
        </Model.TypedElement.type>
        </Model.AssociationEnd>
    </Model.Namespace.contents>
    </Model.Association>
    - <!--
    ASOCIACION: se-refiere-a
    -->
    <Model.Association xmi.id="a3D74DF8003CA">
    <Model.ModelElement.name>se-refiere-a</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="true" />
    <Model.GeneralizableElement.isLeaf xmi.value="true" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Association.isDerived xmi.value="false" />
    <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    </Model.ModelElement.container>
    <Model.Namespace.contents>
    - <!--
    FINAL DE ASOCIACIÓN: elemento
    -->
    <Model.AssociationEnd xmi.id="a3D74DF8A0310">

        <Model.ModelElement.name>elemento</Model.ModelEI
        element.name>
        <Model.ModelElement.annotation />
        <Model.AssociationEnd.isNavigable xmi.value="true" />
        <Model.AssociationEnd.aggregation xmi.value="none" />
        <Model.AssociationEnd.multiplicity>

```

```

        <XMI.field>1</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
- <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D74DF8003CA" />
</Model.ModelElement.container>
- <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3D74DD9A0122" />
</Model.TypedElement.type>
</Model.AssociationEnd>
- <!--
FINAL DE ASOCIACIÓN: restriccion
-->
- <Model.AssociationEnd xmi.id="a3D74DF8A0311">

    <Model.ModelElement.name>restriccion</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
- <Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
- <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D74DF8003CA" />
</Model.ModelElement.container>
- <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3D74DE1201FE" />
</Model.TypedElement.type>
</Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
ASOCIACION: esta-restringido-por
-->
- <Model.Association xmi.id="a3D85B2A101F4">
    <Model.ModelElement.name>esta-restringido-
por</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="true" />
    <Model.GeneralizableElement.isLeaf xmi.value="true" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Association.isDerived xmi.value="false" />
- <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
</Model.ModelElement.container>
- <Model.Namespace.contents>
    - <!--
    FINAL DE ASOCIACIÓN: restriccion
    -->
    - <Model.AssociationEnd xmi.id="a3D85B2A2037C">

        <Model.ModelElement.name>restriccion</Model.ModelElement.name>
        <Model.ModelElement.annotation />
        <Model.AssociationEnd.isNavigable xmi.value="true" />
        <Model.AssociationEnd.aggregation xmi.value="none" />
- <Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>true</XMI.field>
    </Model.AssociationEnd.multiplicity>

```

```

        <Model.AssociationEnd.isChangeable xmi.value="true" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D85B2A101F4" />
    - <Model.ModelElement.container>
    - <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D74DE1201FE" />
    - </Model.TypedElement.type>
    </Model.AssociationEnd>
- <!--
FINAL DE ASOCIACIÓN: proyecto
-->
- <Model.AssociationEnd xmi.id="a3D85B2A2037D">
    <Model.ModelElement.name>proyecto</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
    - <Model.AssociationEnd.multiplicity>
        <XMI.field>1</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
    - </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D85B2A101F4" />
    - </Model.ModelElement.container>
    - <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D74DD4F019D" />
    - </Model.TypedElement.type>
    </Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
ASOCIACION: contiene
-->
- <Model.Association xmi.id="a3D85B30C0383">
    <Model.ModelElement.name>contiene</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="true" />
    <Model.GeneralizableElement.isLeaf xmi.value="true" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Association.isDerived xmi.value="false" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    - </Model.ModelElement.container>
    - <Model.Namespace.contents>
    - <!--
FINAL DE ASOCIACIÓN: compuesto
-->
- <Model.AssociationEnd xmi.id="a3D85B31002DE">
    <Model.ModelElement.name>compuesto</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="true" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
    - <Model.AssociationEnd.multiplicity>
        <XMI.field>0</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
    - </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D85B30C0383" />
    - </Model.ModelElement.container>
    - <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3A241DFC001E" />

```

```

        </Model.TypedElement.type>
    </Model.AssociationEnd>
- <!--
    FINAL DE ASOCIACIÓN: componente
-->
- <Model.AssociationEnd xmi.id="a3D85B31002F2">
    <Model.ModelElement.name>componente</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="shared" />
- <Model.AssociationEnd.multiplicity>
    <XMI.field>0</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
- <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D85B30C0383" />
    </Model.ModelElement.container>
- <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3A241DFC001E" />
    </Model.TypedElement.type>
</Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
    ASOCIACION: utiliza
-->
- <Model.Association xmi.id="a3D85C8070161">
    <Model.ModelElement.name>utiliza</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="true" />
    <Model.GeneralizableElement.isLeaf xmi.value="true" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Association.isDerived xmi.value="false" />
- <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    </Model.ModelElement.container>
- <Model.Namespace.contents>
    - <!--
        FINAL DE ASOCIACIÓN: procedimiento_utilizado
    -->
- <Model.AssociationEnd xmi.id="a3D85C809013C">
    <Model.ModelElement.name>procedimiento_utilizado</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="true" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
- <Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
- <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D85C8070161" />
    </Model.ModelElement.container>
- <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3D85C7DF024C" />
    </Model.TypedElement.type>
</Model.AssociationEnd>
- <!--
    FINAL DE ASOCIACIÓN: actividad
-->
- <Model.AssociationEnd xmi.id="a3D85C809013D">

```



```

        <Model.ModelElement.name>actividad</Model.ModelEle
        ment.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
<Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
</Model.AssociationEnd.multiplicity>
<Model.AssociationEnd.isChangeable xmi.value="true" />
<Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D85C8070161" />
</Model.ModelElement.container>
<Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3A241DFC0024" />
</Model.TypedElement.type>
</Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
ASOCIACION: puede-utilizar
-->
<Model.Association xmi.id="a3D85C80B035C">
    <Model.ModelElement.name>puede-utilizar</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="true" />
    <Model.GeneralizableElement.isLeaf xmi.value="true" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Association.isDerived xmi.value="false" />
<Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
</Model.ModelElement.container>
<Model.Namespace.contents>
- <!--
    FINAL DE ASOCIACIÓN: procedimiento
-->
<Model.AssociationEnd xmi.id="a3D85C80F0086">

    <Model.ModelElement.name>procedimiento</Model.Mo
    delElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="true" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
<Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
</Model.AssociationEnd.multiplicity>
<Model.AssociationEnd.isChangeable xmi.value="true" />
<Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D85C80B035C" />
</Model.ModelElement.container>
<Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3D85C7DF024C" />
</Model.TypedElement.type>
</Model.AssociationEnd>
- <!--
    FINAL DE ASOCIACIÓN: actividad
-->
<Model.AssociationEnd xmi.id="a3D85C80F0090">

    <Model.ModelElement.name>actividad</Model.ModelEle
    ment.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />

```

```

- <Model.AssociationEnd.multiplicity>
  <XML.field>1</XML.field>
  <XML.field>-1</XML.field>
  <XML.field>>false</XML.field>
  <XML.field>>true</XML.field>
</Model.AssociationEnd.multiplicity>
<Model.AssociationEnd.isChangeable xmi.value="true" />
- <Model.ModelElement.container>
  <Model.Namespace xmi.idref="a3D85C80B035C" />
</Model.ModelElement.container>
- <Model.TypedElement.type>
  <Model.Classifier xmi.idref="a3A241DFC0024" />
</Model.TypedElement.type>
</Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
ASOCIACION: modifica
-->
- <Model.Association xmi.id="a3D85C8680184">
  <Model.ModelElement.name>modifica</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.GeneralizableElement.isRoot xmi.value="true" />
  <Model.GeneralizableElement.isLeaf xmi.value="true" />
  <Model.GeneralizableElement.isAbstract xmi.value="false" />
  <Model.GeneralizableElement.visibility xmi.value="public_vis" />
  <Model.Association.isDerived xmi.value="false" />
- <Model.ModelElement.container>
  <Model.Namespace xmi.idref="a3A23FAFC03CA" />
</Model.ModelElement.container>
- <Model.Namespace.contents>
  - <!--
  FINAL DE ASOCIACIÓN: artefacto
  -->
  - <Model.AssociationEnd xmi.id="a3D85C86C0111">
    <Model.ModelElement.name>artefacto</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="true" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
  - <Model.AssociationEnd.multiplicity>
    <XML.field>0</XML.field>
    <XML.field>-1</XML.field>
    <XML.field>>false</XML.field>
    <XML.field>>true</XML.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
  - <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D85C8680184" />
    </Model.ModelElement.container>
  - <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3A241DFC001E" />
    </Model.TypedElement.type>
  </Model.AssociationEnd>
  - <!--
  FINAL DE ASOCIACIÓN: procedimiento
  -->
  - <Model.AssociationEnd xmi.id="a3D85C86C011B">
    <Model.ModelElement.name>procedimiento</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
  - <Model.AssociationEnd.multiplicity>
    <XML.field>1</XML.field>
    <XML.field>-1</XML.field>
    <XML.field>>false</XML.field>
    <XML.field>>true</XML.field>
    </Model.AssociationEnd.multiplicity>

```

```

        <Model.AssociationEnd.isChangeable xmi.value="true" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D85C8680184" />
    - </Model.ModelElement.container>
    - <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D85C7DF024C" />
    - </Model.TypedElement.type>
    </Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
ASOCIACION: utiliza
-->
- <Model.Association xmi.id="a3D85C92C00A0">
    <Model.ModelElement.name>utiliza</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="true" />
    <Model.GeneralizableElement.isLeaf xmi.value="true" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Association.isDerived xmi.value="false" />
    - <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    - </Model.ModelElement.container>
    - <Model.Namespace.contents>
        - <!--
        FINAL DE ASOCIACIÓN: recurso
        -->
        - <Model.AssociationEnd xmi.id="a3D85C92D03CC">
            <Model.ModelElement.name>recurso</Model.ModelElement.name>
            <Model.ModelElement.annotation />
            <Model.AssociationEnd.isNavigable xmi.value="true" />
            <Model.AssociationEnd.aggregation xmi.value="none" />
            - <Model.AssociationEnd.multiplicity>
                <XMI.field>0</XMI.field>
                <XMI.field>-1</XMI.field>
                <XMI.field>>false</XMI.field>
                <XMI.field>true</XMI.field>
            - </Model.AssociationEnd.multiplicity>
            <Model.AssociationEnd.isChangeable xmi.value="true" />
            - <Model.ModelElement.container>
                <Model.Namespace xmi.idref="a3D85C92C00A0" />
            - </Model.ModelElement.container>
            - <Model.TypedElement.type>
                <Model.Classifier xmi.idref="a3A241DFC0029" />
            - </Model.TypedElement.type>
            </Model.AssociationEnd>
        - <!--
        FINAL DE ASOCIACIÓN: actividad
        -->
        - <Model.AssociationEnd xmi.id="a3D85C92D03D6">
            <Model.ModelElement.name>actividad</Model.ModelElement.name>
            <Model.ModelElement.annotation />
            <Model.AssociationEnd.isNavigable xmi.value="false" />
            <Model.AssociationEnd.aggregation xmi.value="none" />
            - <Model.AssociationEnd.multiplicity>
                <XMI.field>1</XMI.field>
                <XMI.field>-1</XMI.field>
                <XMI.field>>false</XMI.field>
                <XMI.field>true</XMI.field>
            - </Model.AssociationEnd.multiplicity>
            <Model.AssociationEnd.isChangeable xmi.value="true" />
            - <Model.ModelElement.container>
                <Model.Namespace xmi.idref="a3D85C92C00A0" />
            - </Model.ModelElement.container>
            - <Model.TypedElement.type>
                <Model.Classifier xmi.idref="a3A241DFC0024" />

```

```

        </Model.TypedElement.type>
    </Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
ASOCIACION: es-propietario-de
-->
<Model.Association xmi.id="a3D85C9B90073">
    <Model.ModelElement.name>es-propietario-
de</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="true" />
    <Model.GeneralizableElement.isLeaf xmi.value="true" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Association.isDerived xmi.value="false" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    </Model.ModelElement.container>
    <Model.Namespace.contents>
        - <!--
        FINAL DE ASOCIACIÓN: recurso
        -->
        <Model.AssociationEnd xmi.id="a3D85C9BA0197">
            <Model.ModelElement.name>recurso</Model.ModelEle
ment.name>
            <Model.ModelElement.annotation />
            <Model.AssociationEnd.isNavigable xmi.value="true" />
            <Model.AssociationEnd.aggregation xmi.value="none" />
            <Model.AssociationEnd.multiplicity>
                <XMI.field>0</XMI.field>
                <XMI.field>-1</XMI.field>
                <XMI.field>>false</XMI.field>
                <XMI.field>>true</XMI.field>
            </Model.AssociationEnd.multiplicity>
            <Model.AssociationEnd.isChangeable xmi.value="true" />
            <Model.ModelElement.container>
                <Model.Namespace xmi.idref="a3D85C9B90073" />
            </Model.ModelElement.container>
            <Model.TypedElement.type>
                <Model.Classifier xmi.idref="a3A241DFC0029" />
            </Model.TypedElement.type>
        </Model.AssociationEnd>
        - <!--
        FINAL DE ASOCIACIÓN: agente
        -->
        <Model.AssociationEnd xmi.id="a3D85C9BA01A1">
            <Model.ModelElement.name>agente</Model.ModelElem
ent.name>
            <Model.ModelElement.annotation />
            <Model.AssociationEnd.isNavigable xmi.value="false" />
            <Model.AssociationEnd.aggregation xmi.value="none" />
            <Model.AssociationEnd.multiplicity>
                <XMI.field>0</XMI.field>
                <XMI.field>1</XMI.field>
                <XMI.field>>false</XMI.field>
                <XMI.field>>false</XMI.field>
            </Model.AssociationEnd.multiplicity>
            <Model.AssociationEnd.isChangeable xmi.value="true" />
            <Model.ModelElement.container>
                <Model.Namespace xmi.idref="a3D85C9B90073" />
            </Model.ModelElement.container>
            <Model.TypedElement.type>
                <Model.Classifier xmi.idref="a3D74DE040262" />
            </Model.TypedElement.type>
        </Model.AssociationEnd>
    </Model.Namespace.contents>
</Model.Association>
- <!--

```

ASOCIACION: es-responsable-de

```
-->
<Model.Association xmi.id="a3D85CA2C0019">
  <Model.ModelElement.name>es-responsable-
    de</Model.ModelElement.name>
  <Model.ModelElement.annotation />
  <Model.GeneralizableElement.isRoot xmi.value="true" />
  <Model.GeneralizableElement.isLeaf xmi.value="true" />
  <Model.GeneralizableElement.isAbstract xmi.value="false" />
  <Model.GeneralizableElement.visibility xmi.value="public_vis" />
  <Model.Association.isDerived xmi.value="false" />
</Model.ModelElement.container>
  <Model.Namespace xmi.idref="a3A23FAFC03CA" />
</Model.ModelElement.container>
<Model.Namespace.contents>
  - <!--
    FINAL DE ASOCIACIÓN: actividad
  -->
  <Model.AssociationEnd xmi.id="a3D85CA2D0060">
    <Model.ModelElement.name>actividad</Model.ModelEle
      ment.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="true" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
  <Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
  </Model.AssociationEnd.multiplicity>
  <Model.AssociationEnd.isChangeable xmi.value="true" />
</Model.ModelElement.container>
  <Model.Namespace xmi.idref="a3D85CA2C0019" />
</Model.ModelElement.container>
  <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3A241DFC0024" />
  </Model.TypedElement.type>
</Model.AssociationEnd>
  - <!--
    FINAL DE ASOCIACIÓN: rol
  -->
  <Model.AssociationEnd xmi.id="a3D85CA2D0074">
    <Model.ModelElement.name>rol</Model.ModelElement.
      name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
  <Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>false</XMI.field>
  </Model.AssociationEnd.multiplicity>
  <Model.AssociationEnd.isChangeable xmi.value="true" />
</Model.ModelElement.container>
  <Model.Namespace xmi.idref="a3D85CA2C0019" />
</Model.ModelElement.container>
  <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3D85C9F200AF" />
  </Model.TypedElement.type>
</Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
  - <!--
    ASOCIACION: desempeña
  -->
  <Model.Association xmi.id="a3D85CA740222">
    <Model.ModelElement.name>desempeña</Model.ModelElement.name>
    <Model.ModelElement.annotation />
```

```

<Model.GeneralizableElement.isRoot xmi.value="true" />
<Model.GeneralizableElement.isLeaf xmi.value="true" />
<Model.GeneralizableElement.isAbstract xmi.value="false" />
<Model.GeneralizableElement.visibility xmi.value="public_vis" />
<Model.Association.isDerived xmi.value="false" />
<Model.ModelElement.container>
  <Model.Namespace xmi.idref="a3A23FAFC03CA" />
</Model.ModelElement.container>
<Model.Namespace.contents>
  - <!--
    FINAL DE ASOCIACIÓN: rol
    -->
  <Model.AssociationEnd xmi.id="a3D85CA7600BC">

    <Model.ModelElement.name>rol</Model.ModelElement.
    name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="true" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
  <Model.AssociationEnd.multiplicity>
    <XMI.field>0</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
  </Model.AssociationEnd.multiplicity>
  <Model.AssociationEnd.isChangeable xmi.value="true" />
  <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D85CA740222" />
  </Model.ModelElement.container>
  <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3D85C9F200AF" />
  </Model.TypedElement.type>
</Model.AssociationEnd>
- <!--
  FINAL DE ASOCIACIÓN: agente
  -->
  <Model.AssociationEnd xmi.id="a3D85CA7600BD">

    <Model.ModelElement.name>agente</Model.ModelElem
    ent.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
  <Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>>true</XMI.field>
  </Model.AssociationEnd.multiplicity>
  <Model.AssociationEnd.isChangeable xmi.value="true" />
  <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D85CA740222" />
  </Model.ModelElement.container>
  <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3D74DE040262" />
  </Model.TypedElement.type>
</Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
  ASOCIACION: realiza
  -->
  <Model.Association xmi.id="a3D85CAB80212">
    <Model.ModelElement.name>realiza</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="true" />
    <Model.GeneralizableElement.isLeaf xmi.value="true" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Association.isDerived xmi.value="false" />
  <Model.ModelElement.container>

```

```

        <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    </Model.ModelElement.container>
<!--
    FINAL DE ASOCIACIÓN: actividad
-->
<Model.AssociationEnd xmi.id="a3D85CABB028E">
    <Model.ModelElement.name>actividad</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="true" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
    <Model.AssociationEnd.multiplicity>
        <XMI.field>0</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D85CAB80212" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3A241DFC0024" />
    </Model.TypedElement.type>
</Model.AssociationEnd>
<!--
    FINAL DE ASOCIACIÓN: agente
-->
<Model.AssociationEnd xmi.id="a3D85CABB02AC">
    <Model.ModelElement.name>agente</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
    <Model.AssociationEnd.multiplicity>
        <XMI.field>0</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D85CAB80212" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3D74DE040262" />
    </Model.TypedElement.type>
</Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
<!--
    ASOCIACION: es-entrada-de
-->
<Model.Association xmi.id="a3D85C68B03B3">
    <Model.ModelElement.name>es-entrada-de</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="true" />
    <Model.GeneralizableElement.isLeaf xmi.value="true" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Association.isDerived xmi.value="false" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    </Model.ModelElement.container>
</Model.Namespace.contents>
<!--
    FINAL DE ASOCIACIÓN: actividad_de_entrada
-->

```

```

- <Model.AssociationEnd xmi.id="a3D85C68E0168">
    <Model.ModelElement.name>actividad_de_entrada</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="true" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
- <Model.AssociationEnd.multiplicity>
    <XMI.field>1</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>true</XMI.field>
</Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
- <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D85C68B03B3" />
</Model.ModelElement.container>
- <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3A241DFC0024" />
</Model.TypedElement.type>
</Model.AssociationEnd>
- <!--
FINAL DE ASOCIACIÓN: artefacto
-->
- <Model.AssociationEnd xmi.id="a3D85C68E017C">
    <Model.ModelElement.name>artefacto</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
- <Model.AssociationEnd.multiplicity>
    <XMI.field>0</XMI.field>
    <XMI.field>-1</XMI.field>
    <XMI.field>>false</XMI.field>
    <XMI.field>true</XMI.field>
</Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
- <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3D85C68B03B3" />
</Model.ModelElement.container>
- <Model.TypedElement.type>
    <Model.Classifier xmi.idref="a3A241DFC001E" />
</Model.TypedElement.type>
</Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
ASOCIACION: es-salida-de
-->
- <Model.Association xmi.id="a3D85C69101C6">
    <Model.ModelElement.name>es-salida-de</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="true" />
    <Model.GeneralizableElement.isLeaf xmi.value="true" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.Association.isDerived xmi.value="false" />
- <Model.ModelElement.container>
    <Model.Namespace xmi.idref="a3A23FAFC03CA" />
</Model.ModelElement.container>
- <Model.Namespace.contents>
    - <!--
    FINAL DE ASOCIACIÓN: actividad_de_salida
    -->
- <Model.AssociationEnd xmi.id="a3D85C69403BF">
    <Model.ModelElement.name>actividad_de_salida</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="true" />

```



```

        <Model.AssociationEnd.aggregation xmi.value="none" />
    <Model.AssociationEnd.multiplicity>
        <XMI.field>1</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D85C69101C6" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3A241DFC0024" />
    </Model.TypedElement.type>
</Model.AssociationEnd>
- <!--
FINAL DE ASOCIACIÓN: artefacto
-->
<Model.AssociationEnd xmi.id="a3D85C69403D3">
    <Model.ModelElement.name>artefacto</Model.ModelEle
ment.name>
    <Model.ModelElement.annotation />
    <Model.AssociationEnd.isNavigable xmi.value="false" />
    <Model.AssociationEnd.aggregation xmi.value="none" />
    <Model.AssociationEnd.multiplicity>
        <XMI.field>0</XMI.field>
        <XMI.field>-1</XMI.field>
        <XMI.field>>false</XMI.field>
        <XMI.field>>true</XMI.field>
    </Model.AssociationEnd.multiplicity>
    <Model.AssociationEnd.isChangeable xmi.value="true" />
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3D85C69101C6" />
    </Model.ModelElement.container>
    <Model.TypedElement.type>
        <Model.Classifier xmi.idref="a3A241DFC001E" />
    </Model.TypedElement.type>
</Model.AssociationEnd>
</Model.Namespace.contents>
</Model.Association>
- <!--
DATA TYPE: String
-->
<Model.DataType xmi.id="G.49">
    <Model.ModelElement.name>String</Model.ModelElement.name>
    <Model.ModelElement.annotation />
    <Model.GeneralizableElement.isRoot xmi.value="false" />
    <Model.GeneralizableElement.isLeaf xmi.value="false" />
    <Model.GeneralizableElement.isAbstract xmi.value="false" />
    <Model.GeneralizableElement.visibility xmi.value="public_vis" />
    <Model.DataType.typeCode>
        <XMI.CorbaTypeCode>
            <XMI.CorbaTcString xmi.tcLength="0" />
        </XMI.CorbaTypeCode>
    </Model.DataType.typeCode>
    <Model.ModelElement.container>
        <Model.Namespace xmi.idref="a3A23FAFC03CA" />
    </Model.ModelElement.container>
</Model.DataType>
</Model.Namespace.contents>
</Model.Package>
</XMI.content>
</XMI>

```

I.1.2. Documento DTD del Nivel M2.

Este documento ha sido generado, a la vez que el mostrado en el apartado anterior, utilizando la herramienta METAMOD activando la opción “Nombre del DTD a generar” (ver anexo H). Este DTD, que representa el metamodelo del nivel M2, podrá ser utilizado posteriormente para los documentos XMI que representan modelos del nivel M1 como, por ejemplo, el incluido en el apartado I.2; es decir, dichos modelos M1 deberán satisfacer este DTD.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- _____ -->
<!-- XMI es el elemento XML de alto nivel para la transferencia XMI -->
<!-- _____ -->

<!ELEMENT XMI (XMI.header?, XMI.content?, XMI.difference*,
               XMI.extensions*) >

<!ATTLIST XMI
    xmi.version CDATA #FIXED "1.0"
    timestamp CDATA #IMPLIED
    verified (true | false) #IMPLIED >

<!-- _____ -->
<!-- El elemento XMI.header contiene documentacion e identifica -->
<!-- el modelo, metamodelo, y el metamodelo -->
<!-- _____ -->

<!ELEMENT XMI.header (XMI.documentation?, XMI.model*, XMI.metamodel*,
                     XMI.metamodelmodel*, XMI.import*) >

<!-- _____ -->
<!-- Documentacion para la transferencia de datos -->
<!-- _____ -->

<!ELEMENT XMI.documentation (#PCDATA | XMI.owner | XMI.contact |
                             XMI.longDescription | XMI.shortDescription |
                             XMI.exporter | XMI.exporterVersion | XMI.exporterID |
                             XMI.notice)* >

<!ELEMENT XMI.owner ANY >
<!ELEMENT XMI.contact ANY >
<!ELEMENT XMI.longDescription ANY >
<!ELEMENT XMI.shortDescription ANY >
<!ELEMENT XMI.exporter ANY >
<!ELEMENT XMI.exporterVersion ANY >
<!ELEMENT XMI.exporterID ANY >
<!ELEMENT XMI.notice ANY >

<!-- _____ -->
<!-- XMI.element.att define los atributos que debe tener cada -->
<!-- elemento XML que corresponden a una clase del metamodelo -->
<!-- conforme a la especificacion XMI. -->
<!-- _____ -->

<!ENTITY % XMI.element.att
    'xmi.id ID #IMPLIED xmi.label CDATA #IMPLIED xmi.uuid
    CDATA #IMPLIED ' >

<!-- _____ -->
<!-- XMI.element.link.att define los atributos que debe tener cada -->
<!-- elemento XML que corresponden a una clase del metamodelo -->
<!-- para habilitar una funcion como un enlace simple XLINK y tb -->
<!-- como referencia a un constructor de modelo dentro del mismo -->
<!-- archivo XMI. -->
<!-- _____ -->

<!ENTITY % XMI.link.att
    'xmi:link CDATA #IMPLIED
    inline (true | false) #IMPLIED
    actuate (show | user) #IMPLIED
    href CDATA #IMPLIED
```

```

role CDATA #IMPLIED
title CDATA #IMPLIED
show (embed | replace | new) #IMPLIED
behavior CDATA #IMPLIED
xmi.idref IDREF #IMPLIED
xmi.uidref CDATA #IMPLIED' >

<!-- _____ -->
<!-- XMI.model identifica el modelo que se va a transmitir -->
<!-- _____ -->

<!ELEMENT XMI.model ANY >
<!ATTLIST XMI.model
    %XMI.link.att;
    xmi.name CDATA #REQUIRED
    xmi.version CDATA #IMPLIED >

<!-- _____ -->
<!-- XMI.metamodel identifica el metamodelo que se va a transmitir -->
<!-- datos -->
<!-- _____ -->

<!ELEMENT XMI.metamodel ANY >
<!ATTLIST XMI.metamodel
    %XMI.link.att;
    xmi.name CDATA #REQUIRED
    xmi.version CDATA #IMPLIED >

<!-- _____ -->
<!-- XMI.metametamodel identifica el metametamodelo que se va a tran -->
<!-- datos transferidos -->
<!-- _____ -->

<!ELEMENT XMI.metametamodel ANY >
<!ATTLIST XMI.metametamodel
    %XMI.link.att;
    xmi.name CDATA #REQUIRED
    xmi.version CDATA #IMPLIED >

<!-- _____ -->
<!-- XMI.import identifica los metamodelos importados -->
<!-- _____ -->

<!ELEMENT XMI.import ANY >
<!ATTLIST XMI.import %XMI.link.att;
    xmi.name CDATA #REQUIRED
    xmi.version CDATA #IMPLIED >

<!-- _____ -->
<!-- XMI.content contiene los datos a transferir -->
<!-- _____ -->

<!ELEMENT XMI.content ANY >

<!-- _____ -->
<!-- XMI.extensions contiene los datos a transferir que no son -->
<!-- conformes a los metamodelos de la cabecera (header) -->
<!-- _____ -->

<!ELEMENT XMI.extensions ANY >
<!ATTLIST XMI.extensions
    xmi.extender CDATA #REQUIRED >

<!-- _____ -->
<!-- extension contiene informacion relacionada con un constructor -->
<!-- especifico de un modelo que no esta definido en el metamodelo -->
<!-- de la cabecera -->
<!-- _____ -->

<!ELEMENT XMI.extension ANY >
<!ATTLIST XMI.extension
    %XMI.element.att;
    %XMI.link.att;

```

```

xmi.extender CDATA #REQUIRED
xmi.extenderID CDATA #IMPLIED >

<!-- _____ -->
<!-- XMI.difference mantiene los elementos XML representando las -->
<!-- diferencias a un modelo base -->
<!-- _____ -->

<!ELEMENT XMI.difference (XMI.difference | XMI.delete | XMI.add | XMI.replace)* >
<ATTLIST XMI.difference
    %XMI.element.att;
    %XMI.link.att; >

<!-- _____ -->
<!-- XMI.delete representa un proceso de borrado desde un model base -->
<!-- _____ -->

<!ELEMENT XMI.delete EMPTY >
<ATTLIST XMI.delete
    %XMI.element.att;
    %XMI.link.att; >

<!-- _____ -->
<!-- XMI.add representa una adición a un modelo base -->
<!-- _____ -->

<!ELEMENT XMI.add ANY >
<ATTLIST XMI.add
    %XMI.element.att;
    %XMI.link.att;xmi.position CDATA "-1" >

<!-- _____ -->
<!-- XMI.replace representa la sustitución de un constructor del mod-->
<!-- por otro constructor en un modelo base -->
<!-- _____ -->

<!ELEMENT XMI.replace ANY >
<ATTLIST XMI.replace
    %XMI.element.att;
    %XMI.link.att;
    xmi.position CDATA "-1" >

<!-- _____ -->
<!-- XMI.reference se puede usar para hacer referencia a tipos de -->
<!-- datos no definidos en el metamodelo -->
<!-- _____ -->

<!ELEMENT XMI.reference ANY >
<ATTLIST XMI.reference
    %XMI.link.att; >

<!ELEMENT XMI.TypeDefinitions ANY >

<!ELEMENT XMI.field ANY >
<!ELEMENT XMI.seqItem ANY >
<!ELEMENT XMI.enum EMPTY >

<ATTLIST XMI.enum
    xmi.value CDATA #REQUIRED>
<!ELEMENT XMI.any ANY >
<ATTLIST XMI.any
    %XMI.link.att;
    xmi.type CDATA #IMPLIED
    xmi.name CDATA #IMPLIED>

<!ELEMENT XMI.CorbaTypeCode (XMI.CorbaTcEnum | XMI.CorbaTcString |
    XMI.CorbaTcShort |XMI.CorbaTcBoolean |XMI.CorbaTcTypeCode) >

<ATTLIST XMI.CorbaTypeCode
    %XMI.element.att; >

<!ELEMENT XMI.CorbaTcEnum (XMI.CorbaTcEnumLabel) >
<ATTLIST XMI.CorbaTcEnum

```

```

xmi.tcName CDATA #REQUIRED
xmi.tcId CDATA #IMPLIED >

<!ELEMENT XMI.CorbaTcEnumLabel EMPTY >
<!ATTLIST XMI.CorbaTcEnumLabel
    xmi.tcName CDATA #REQUIRED >

<!ELEMENT XMI.CorbaTcString EMPTY >
<!ATTLIST XMI.CorbaTcString
    xmi.tcLength CDATA #REQUIRED >

<!ELEMENT XMI.CorbaTcBoolean EMPTY >
<!ELEMENT XMI.CorbaTcTypeCode EMPTY >
<!ELEMENT XMI.CorbaTcShort EMPTY >

<!-- ***** -->
<!-- FIN DE LOS ELEMENTOS FIJOS DE TODO DTD XMI -->
<!-- ***** -->

<!-- METAMODELO: org.omg.mof.Model -->
<!-- PAQUETE: Basico -->
<!-- CLASE DEL METAMODELO: Artefacto -->

<!ELEMENT Basico.Artefacto.referencia_a_compuesto (Basico.Artefacto)* >

<!ELEMENT Basico.Artefacto.referencia_a_actividad_de_entrada (Basico.Actividad)* >

<!ELEMENT Basico.Artefacto.referencia_a_actividad_de_salida (Basico.Actividad)* >

<!ELEMENT Basico.Artefacto (Basico.Elemento.nombre , Basico.Artefacto.referencia_a_compuesto* ,
Basico.Artefacto.referencia_a_actividad_de_entrada* , Basico.Artefacto.referencia_a_actividad_de_salida*)? >
<!ATTLIST Basico.Artefacto %XML.element.att; %XML.link.att; >

<!-- CLASE DEL METAMODELO: Recurso -->

<!ELEMENT Basico.Recurso (Basico.Elemento.nombre)? >
<!ATTLIST Basico.Recurso %XML.element.att; %XML.link.att; >

<!-- CLASE DEL METAMODELO: Actividad -->
<!ELEMENT Basico.Actividad.nivelabstraccion (#PCDATA | XML.reference)* >

<!ELEMENT Basico.Actividad.referencia_a_procedimiento_utilizado (Basico.Procedimiento)+ >

<!ELEMENT Basico.Actividad.referencia_a_procedimiento (Basico.Procedimiento)+ >

<!ELEMENT Basico.Actividad.referencia_a_recurso (Basico.Recurso)* >

<!ELEMENT Basico.Actividad (Basico.Elemento.nombre ,
Basico.Actividad.nivelabstraccion,Basico.Actividad.referencia_a_procedimiento_utilizado+ ,
Basico.Actividad.referencia_a_procedimiento+ , Basico.Actividad.referencia_a_recurso*)? >
<!ATTLIST Basico.Actividad %XML.element.att; %XML.link.att; >

<!-- CLASE DEL METAMODELO: Proyecto -->
<!ELEMENT Basico.Proyecto.referencia_a_elemento
(Basico.Elemento|Basico.Artefacto|Basico.Recurso|Basico.Actividad|Basico.Agente)+ >

<!ELEMENT Basico.Proyecto.referencia_a_sub-proyecto (Basico.Proyecto)* >

<!ELEMENT Basico.Proyecto.referencia_a_restriccion (Basico.Restriccion)+ >

<!ELEMENT Basico.Proyecto (Basico.Proyecto.referencia_a_elemento+ , Basico.Proyecto.referencia_a_sub-proyecto*
, Basico.Proyecto.referencia_a_restriccion+)? >
<!ATTLIST Basico.Proyecto %XML.element.att; %XML.link.att; >

<!-- CLASE DEL METAMODELO: Elemento -->

<!ELEMENT Basico.Elemento.nombre (#PCDATA | XML.reference)* >
<!ELEMENT Basico.Elemento (Basico.Elemento.nombre)? >
<!ATTLIST Basico.Elemento %XML.element.att; %XML.link.att; >

<!-- CLASE DEL METAMODELO: Agente -->

<!ELEMENT Basico.Agente.referencia_a_recurso (Basico.Recurso)* >

```

```
<!ELEMENT Basico.Agente.referencia_a_rol (Basico.Rol)* >

<!ELEMENT Basico.Agente.referencia_a_actividad (Basico.Actividad)* >

<!ELEMENT Basico.Agente (Basico.Elemento.nombre,Basico.Agente.referencia_a_recurso*,
Basico.Agente.referencia_a_rol*, Basico.Agente.referencia_a_actividad*)? >
<!ATTLIST Basico.Agente %XML.element.att; %XML.link.att; >

<!--          CLASE DEL METAMODELO: Restriccion          -->
<!ELEMENT Basico.Restriccion.referencia_a_elemento
(Basico.Elemento|Basico.Artefacto|Basico.Recurso|Basico.Actividad|Basico.Agente)+ >

<!ELEMENT Basico.Restriccion (Basico.Restriccion.referencia_a_elemento+)? >
<!ATTLIST Basico.Restriccion %XML.element.att; %XML.link.att; >

<!--          CLASE DEL METAMODELO: Procedimiento          -->
<!ELEMENT Basico.Procedimiento.referencia_a_artefacto (Basico.Artefacto)* >

<!ELEMENT Basico.Procedimiento (Basico.Elemento.nombre, Basico.Procedimiento.referencia_a_artefacto*)? >
<!ATTLIST Basico.Procedimiento %XML.element.att; %XML.link.att; >

<!--          CLASE DEL METAMODELO: Rol          -->

<!ELEMENT Basico.Rol.referencia_a_actividad (Basico.Actividad)+ >

<!ELEMENT Basico.Rol (Basico.Elemento.nombre, Basico.Rol.referencia_a_actividad+)? >
<!ATTLIST Basico.Rol %XML.element.att; %XML.link.att; >

<!--          CONTENIDO DEL PAQUETE Basico          -->

<!ELEMENT Basico
((Basico.Artefacto|Basico.Recurso|Basico.Actividad|Basico.Proyecto|Basico.Elemento|Basico.Agente|
Basico.Restriccion|Basico.Procedimiento|Basico.Rol)*) >
<!ATTLIST Basico %XML.element.att; %XML.link.att; >
```

I.2. Documentos para Correspondencias M2-M1.

A continuación se muestra el documento XML para las correspondencias entre los niveles M2 y M1. En este caso de ejemplo, el documento establece asociaciones entre objetos pertenecientes al metamodelo M2 mostrado anteriormente (apartado I.1) y sus instancias de nivel M1 pertenecientes al modelo de actividades de mantenimiento no planificable de la metodología MANTEMA (anexo G).

Este documento ha sido generado con la herramienta METAMOD de forma similar al del nivel M2 mostrado en el apartado I.1.1, pero asignando el valor 1 a la propiedad nivel y eligiendo como “Referencia a DTD” el archivo mostrado en el apartado I.1.2. En el anexo H se explican los detalles del uso de estos parámetros al invocar los servicios del gestor del repositorio, bien desde METAMOD o bien desde cualquier otra herramienta de metamodelado.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE XMI (View Source for full doctype...)>
<XMI xmi.version="1.0">
  <XMI.header>
    <XMI.documentation>
      <XMI.owner>MANTIS-METAMOD</XMI.owner>
      <XMI.contact>Grupo Alarcos</XMI.contact>
      <XMI.longDescription>SubModelo de MANTEMA: Mantenimiento no
        Planificable</XMI.longDescription>
      <XMI.shortDescription>MODELO NIVEL 2-1</XMI.shortDescription>
      <XMI.exporter>MANTIS-METAMOD</XMI.exporter>
      <XMI.exporterVersion>1.0</XMI.exporterVersion>
    </XMI.documentation>
    <XMI.model xmi.name="Modelo Mantema (Mantenimiento no Planificable)" xmi.version="1.0" />
  </XMI.header>
</XMI>
```

```

<XMI.metamodel xmi.name="MetaProcesos Software" xmi.version="1.0" />
<XMI.metamodel xmi.name="MOF" xmi.version="1.3" />
</XMI.header>
<XMI.content>
- <!--
  MODELO: Modelo Mantema (Mantenimiento no Planificable)
-->
- <!--
  INSTANCIAS DEL PAQUETE: MetaMantema
-->
- <!--
  INSTANCIAS DE LA CLASE: Actividad
-->
- <Basico.Actividad xmi.id="a44F4690DA09011D5BD4C086BBF663F78">
  <Basico.Elemento.nombre>Analisis Error</Basico.Elemento.nombre>
  <Basico.Actividad.nivelabstraccion>2</Basico.Actividad.nivelabstraccion>
  - <Basico.Actividad.referencia_a_procedimiento_utilizado>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F405" />
    </Basico.Actividad.referencia_a_procedimiento_utilizado>
  - <Basico.Actividad.referencia_a_procedimiento>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F405" />
    </Basico.Actividad.referencia_a_procedimiento>
  </Basico.Actividad>
- <Basico.Actividad xmi.id="a44F4690DA09011D5BD4C086BBF663F79">
  <Basico.Elemento.nombre>Investigar y analizar causas</Basico.Elemento.nombre>
  <Basico.Actividad.nivelabstraccion>2</Basico.Actividad.nivelabstraccion>
  - <Basico.Actividad.referencia_a_procedimiento_utilizado>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F400" />
    </Basico.Actividad.referencia_a_procedimiento_utilizado>
  - <Basico.Actividad.referencia_a_procedimiento>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F400" />
    </Basico.Actividad.referencia_a_procedimiento>
  </Basico.Actividad>
- <Basico.Actividad xmi.id="a44F4690DA09011D5BD4C086BBF663F80">
  <Basico.Elemento.nombre>Intervencion correctiva urgente</Basico.Elemento.nombre>
  <Basico.Actividad.nivelabstraccion>2</Basico.Actividad.nivelabstraccion>
  - <Basico.Actividad.referencia_a_procedimiento_utilizado>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F406" />
    </Basico.Actividad.referencia_a_procedimiento_utilizado>
  - <Basico.Actividad.referencia_a_procedimiento>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F406" />
    </Basico.Actividad.referencia_a_procedimiento>
  </Basico.Actividad>
- <Basico.Actividad xmi.id="a44F4690DA09011D5BD4C086BBF663F81">
  <Basico.Elemento.nombre>Realizar acciones correctivas</Basico.Elemento.nombre>
  <Basico.Actividad.nivelabstraccion>2</Basico.Actividad.nivelabstraccion>
  - <Basico.Actividad.referencia_a_procedimiento_utilizado>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F401" />
    </Basico.Actividad.referencia_a_procedimiento_utilizado>
  - <Basico.Actividad.referencia_a_procedimiento>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F408" />
    </Basico.Actividad.referencia_a_procedimiento>
  </Basico.Actividad>
- <Basico.Actividad xmi.id="a44F4690DA09011D5BD4C086BBF663F82">
  <Basico.Elemento.nombre>Cumplimentar documentacion</Basico.Elemento.nombre>
  <Basico.Actividad.nivelabstraccion>2</Basico.Actividad.nivelabstraccion>
  - <Basico.Actividad.referencia_a_procedimiento_utilizado>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F402" />
    </Basico.Actividad.referencia_a_procedimiento_utilizado>
  - <Basico.Actividad.referencia_a_procedimiento>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F402" />
    </Basico.Actividad.referencia_a_procedimiento>
  </Basico.Actividad>
- <Basico.Actividad xmi.id="a44F4690DA09011D5BD4C086BBF663F83">
  <Basico.Elemento.nombre>Ejecutar pruebas unitarias</Basico.Elemento.nombre>
  <Basico.Actividad.nivelabstraccion>2</Basico.Actividad.nivelabstraccion>
  - <Basico.Actividad.referencia_a_procedimiento_utilizado>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F403" />
    </Basico.Actividad.referencia_a_procedimiento_utilizado>
  - <Basico.Actividad.referencia_a_procedimiento>
    <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F409" />
    </Basico.Actividad.referencia_a_procedimiento>
  </Basico.Actividad>

```

```

        </Basico.Actividad.referencia_a_procedimiento>
    </Basico.Actividad>
- <Basico.Actividad xmi.id="a44F4690DA09011D5BD4C086BBF663F84">
    <Basico.Elemento.nombre>Cierre intervencion</Basico.Elemento.nombre>
    <Basico.Actividad.nivelabstraccion>2</Basico.Actividad.nivelabstraccion>
    - <Basico.Actividad.referencia_a_procedimiento_utilizado>
        <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F407" />
    </Basico.Actividad.referencia_a_procedimiento_utilizado>
    - <Basico.Actividad.referencia_a_procedimiento>
        <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F407" />
    </Basico.Actividad.referencia_a_procedimiento>
    </Basico.Actividad>
- <Basico.Actividad xmi.id="a44F4690DA09011D5BD4C086BBF663F85">
    <Basico.Elemento.nombre>Pasar a produccion</Basico.Elemento.nombre>
    <Basico.Actividad.nivelabstraccion>2</Basico.Actividad.nivelabstraccion>
    - <Basico.Actividad.referencia_a_procedimiento_utilizado>
        <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F404" />
    </Basico.Actividad.referencia_a_procedimiento_utilizado>
    - <Basico.Actividad.referencia_a_procedimiento>
        <Basico.Procedimiento xmi.idref="a44F4690CA09011D5BD4C086BBF66F404" />
    </Basico.Actividad.referencia_a_procedimiento>
    </Basico.Actividad>
- <!--
INSTANCIAS DE LA CLASE: Artefacto
-->
- <Basico.Artefacto xmi.id="a44F4690CA09011D5BD4C086BBF66F99">
    <Basico.Elemento.nombre>Conjunto de Elementos Software a
    Corregir</Basico.Elemento.nombre>
    - <Basico.Artefacto.referencia_a_actividad_de_entrada>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F81" />
    </Basico.Artefacto.referencia_a_actividad_de_entrada>
    - <Basico.Artefacto.referencia_a_actividad_de_salida>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F79" />
    </Basico.Artefacto.referencia_a_actividad_de_salida>
    </Basico.Artefacto>
- <Basico.Artefacto xmi.id="a44F4690CA09011D5BD4C086BBF66F100">
    <Basico.Elemento.nombre>Producto Software en Explotacion con Error Bloqueante o
    Critico</Basico.Elemento.nombre>
    - <Basico.Artefacto.referencia_a_actividad_de_entrada>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F79" />
    </Basico.Artefacto.referencia_a_actividad_de_entrada>
    </Basico.Artefacto>
- <Basico.Artefacto xmi.id="a44F4690CA09011D5BD4C086BBF66F101">
    <Basico.Elemento.nombre>Elementos Software Corregidos</Basico.Elemento.nombre>
    - <Basico.Artefacto.referencia_a_actividad_de_entrada>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F83" />
    </Basico.Artefacto.referencia_a_actividad_de_entrada>
    </Basico.Artefacto>
- <Basico.Artefacto xmi.id="a44F4690CA09011D5BD4C086BBF66F102">
    <Basico.Elemento.nombre>Elementos Software Antiguos (con errores
    visibles)</Basico.Elemento.nombre>
    - <Basico.Artefacto.referencia_a_actividad_de_entrada>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F82" />
    </Basico.Artefacto.referencia_a_actividad_de_entrada>
    </Basico.Artefacto>
- <Basico.Artefacto xmi.id="a44F4690CA09011D5BD4C086BBF66F103">
    <Basico.Elemento.nombre>Casos de Prueba</Basico.Elemento.nombre>
    - <Basico.Artefacto.referencia_a_actividad_de_entrada>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F83" />
    </Basico.Artefacto.referencia_a_actividad_de_entrada>
    </Basico.Artefacto>
- <Basico.Artefacto xmi.id="a44F4690CA09011D5BD4C086BBF66F104">
    <Basico.Elemento.nombre>Elementos Software Corregidos y
    Probados</Basico.Elemento.nombre>
    - <Basico.Artefacto.referencia_a_actividad_de_entrada>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F85" />
    </Basico.Artefacto.referencia_a_actividad_de_entrada>
    - <Basico.Artefacto.referencia_a_actividad_de_salida>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F83" />
    </Basico.Artefacto.referencia_a_actividad_de_salida>
    </Basico.Artefacto>
- <Basico.Artefacto xmi.id="a44F4690CA09011D5BD4C086BBF66F105">

```



```

        <Basico.Elemento.nombre>Documentacion de las acciones correctivas
        realizadas</Basico.Elemento.nombre>
    - <Basico.Artefacto.referencia_a_actividad_de_salida>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F82" />
    </Basico.Artefacto.referencia_a_actividad_de_salida>
</Basico.Artefacto>
- <Basico.Artefacto xmi.id="a44F4690CA09011D5BD4C086BBF66F106">
    <Basico.Elemento.nombre>Documentacion de las pruebas unitarias
    realizadas</Basico.Elemento.nombre>
    - <Basico.Artefacto.referencia_a_actividad_de_salida>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F83" />
    <Basico.Artefacto.referencia_a_actividad_de_salida>
</Basico.Artefacto>
- <Basico.Artefacto xmi.id="a44F4690CA09011D5BD4C086BBF66F107">
    <Basico.Elemento.nombre>Producto Software en Explotacion
    Corregido</Basico.Elemento.nombre>
    - <Basico.Artefacto.referencia_a_actividad_de_salida>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F85" />
    <Basico.Artefacto.referencia_a_actividad_de_salida>
</Basico.Artefacto>
- <Basico.Artefacto xmi.id="a44F4690CA09011D5BD4C086BBF66F108">
    <Basico.Elemento.nombre>Elementos Software Corregidos y
    Probados</Basico.Elemento.nombre>
    - <Basico.Artefacto.referencia_a_actividad_de_entrada>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F85" />
    <Basico.Artefacto.referencia_a_actividad_de_entrada>
    - <Basico.Artefacto.referencia_a_actividad_de_salida>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F83" />
    <Basico.Artefacto.referencia_a_actividad_de_salida>
</Basico.Artefacto>
- <!--
INSTANCIAS DE LA CLASE: Rol
-->
- <Basico.Rol xmi.id="a44F4690CA09011D5BD4C086BBF66200">
    <Basico.Elemento.nombre>Equipo de Mantenimiento</Basico.Elemento.nombre>
    - <Basico.Rol.referencia_a_actividad>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F79" />
    <Basico.Rol.referencia_a_actividad>
    - <Basico.Rol.referencia_a_actividad>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F81" />
    <Basico.Rol.referencia_a_actividad>
    - <Basico.Rol.referencia_a_actividad>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F82" />
    <Basico.Rol.referencia_a_actividad>
    - <Basico.Rol.referencia_a_actividad>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F83" />
    <Basico.Rol.referencia_a_actividad>
    - <Basico.Rol.referencia_a_actividad>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F85" />
    <Basico.Rol.referencia_a_actividad>
</Basico.Rol>
- <Basico.Rol xmi.id="a44F4690CA09011D5BD4C086BBF66201">
    <Basico.Elemento.nombre>Usuario</Basico.Elemento.nombre>
    - <Basico.Rol.referencia_a_actividad>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F79" />
    <Basico.Rol.referencia_a_actividad>
    - <Basico.Rol.referencia_a_actividad>
        <Basico.Actividad xmi.idref="a44F4690DA09011D5BD4C086BBF663F85" />
    <Basico.Rol.referencia_a_actividad>
</Basico.Rol>
- <!--
INSTANCIAS DE LA CLASE: Agente
-->
- <Basico.Agente xmi.id="a44F4690CA09011D5BD4C086BBF66300">
    <Basico.Elemento.nombre>Usuario</Basico.Elemento.nombre>
    - <Basico.Agente.referencia_a_rol>
        <Basico.Rol xmi.idref="a44F4690CA09011D5BD4C086BBF66201" />
    <Basico.Agente.referencia_a_rol>
</Basico.Agente>
- <Basico.Agente xmi.id="a44F4690CA09011D5BD4C086BBF66301">
    <Basico.Elemento.nombre>Mantenedor</Basico.Elemento.nombre>
    - <Basico.Agente.referencia_a_rol>

```

```

        <Basico.Rol xmi.idref="a44F4690CA09011D5BD4C086BBF66200" />
    </Basico.Agente.referencia_a_rol>
</Basico.Agente>
- <!--
INSTANCIAS DE LA CLASE: Procedimiento
-->
- <Basico.Procedimiento xmi.id="a44F4690CA09011D5BD4C086BBF66F400">
    <Basico.Elemento.nombre>Procedimiento Investigar y Analizar
    Causas</Basico.Elemento.nombre>
</Basico.Procedimiento>
- <Basico.Procedimiento xmi.id="a44F4690CA09011D5BD4C086BBF66F401">
    <Basico.Elemento.nombre>Realizar Acciones Correctivas</Basico.Elemento.nombre>
</Basico.Procedimiento>
- <Basico.Procedimiento xmi.id="a44F4690CA09011D5BD4C086BBF66F402">
    <Basico.Elemento.nombre>Procedimiento Cumplimentar
    Documentacion</Basico.Elemento.nombre>
</Basico.Procedimiento>
- <Basico.Procedimiento xmi.id="a44F4690CA09011D5BD4C086BBF66F403">
    <Basico.Elemento.nombre>Procedimiento Ejecutar Pruebas
    Unitarias</Basico.Elemento.nombre>
</Basico.Procedimiento>
- <Basico.Procedimiento xmi.id="a44F4690CA09011D5BD4C086BBF66F404">
    <Basico.Elemento.nombre>Procedimiento Pasar a
    Produccion</Basico.Elemento.nombre>
</Basico.Procedimiento>
- <Basico.Procedimiento xmi.id="a44F4690CA09011D5BD4C086BBF66F405">
    <Basico.Elemento.nombre>Procedimiento Analisis del Error</Basico.Elemento.nombre>
</Basico.Procedimiento>
- <Basico.Procedimiento xmi.id="a44F4690CA09011D5BD4C086BBF66F406">
    <Basico.Elemento.nombre>Procedimiento Intervencion Correctiva
    Urgente</Basico.Elemento.nombre>
</Basico.Procedimiento>
- <Basico.Procedimiento xmi.id="a44F4690CA09011D5BD4C086BBF66F407">
    <Basico.Elemento.nombre>Procedimiento Cierre
    Intervencion</Basico.Elemento.nombre>
</Basico.Procedimiento>
- <Basico.Procedimiento xmi.id="a44F4690CA09011D5BD4C086BBF66F408">
    <Basico.Elemento.nombre>Procedimiento Codificacion</Basico.Elemento.nombre>
</Basico.Procedimiento>
- <Basico.Procedimiento xmi.id="a44F4690CA09011D5BD4C086BBF66F409">
    <Basico.Elemento.nombre>Tecnicas de Prueba del Software</Basico.Elemento.nombre>
</Basico.Procedimiento>
</XMI.content>
</XMI>

```

Apéndices.

Lista de Acrónimos.

Referencias Bibliográficas.

1. Lista de Acrónimos.

4GL	Lenguaje de Cuarta Generación
ACID	Atomic, Consistent, Isolation, and Durable
ACL	<i>Agent Communication Language</i>
ACM	<i>Association for Computing Machinery</i>
AIAA	<i>American Institute of Aeronautics and Astronautics</i>
ANS	Acuerdo de Nivel de Servicio
ANSI	<i>American National Standards Institute</i>
API	<i>Application Programming Interface</i>
APSE	<i>Ada Programming Support Environment</i>
APSEC	<i>Asia Pacific Software Engineering Conference</i>
ARIS	<i>Architecture of Integrated Information Systems</i>
ASCII	<i>American Standard Code for Information Interchange</i>
ASE	<i>Annals of Software Engineering</i>
ATIS	<i>A Tool Integration Standard</i>
BPR	<i>Business Process Reengineering</i>
CAIS-A	<i>Common APSE Interface Set</i>
CAISE	<i>Conference on Advanced Information Systems Engineering</i>
CASE	<i>Computer Aided Software Engineering</i>
CCS	<i>Calculus of Communicating Systems</i>
CDIF	<i>CASE Data Interchange Format</i>
CENPRI	Centro Provincial de Informática
CICS	<i>Customer Information Control System</i>
CICYT	Comisión Interministerial de Ciencia y Tecnología
CM3	<i>Corrective Maintenance Maturity Model</i>
CMM	<i>Capability Maturity Model</i>
CMMi	<i>Capability Maturity Model Integration</i>
CNS	Contrato de Nivel de Servicio
COCOMO	<i>Constructive Cost Model</i>
CORBA	<i>Common Object Request Broker Architecture</i>
COTS	<i>Commercial Off-The-Shelf</i>
CPU	<i>Central Processing Unit</i>
CRM	<i>Customer Relations Management</i>
CSCW	<i>Computer Supported Collaborative Work</i>
CSI	Consejo Superior de Informática
CSMR	<i>European Conference on Software Maintenance and Reengineering</i>
CSP	<i>Communicating Sequential Processes</i>
DOM	<i>Document Object Model</i>
DRAE	Diccionario de la Real Academia Española de la Lengua
DTD	<i>Document Type Definition</i>
EBNF	<i>Extended Backus Naur Form</i>
EC	<i>European Commission</i>
ECMA	<i>European Computer Manufacturers Association</i>
EIA	<i>Electronic Industries Alliance</i>
EIA/IS	<i>Electronic Industries Alliance / Interim Standard</i>
EIS	Entornos de Ingeniería del Software
ESA	<i>European Space Agency</i>
ESE	<i>Empirical Software Engineering</i>
ESEC	<i>European Software Engineering Conference</i>
ESP	Entornos de Soporte a Proyectos
EWSPT	<i>European Workshop on Software Process Technology</i>
FASE	<i>International Conference on Fundamental Approaches to Software Engineering</i>
FEDER	Fondo Europeo de Desarrollo Regional
FIPA	<i>Foundation for Intelligent Physical Agents</i>

FRISCO	<i>Framework of Information Systems Concepts</i>
FT	<i>Flujo de Trabajo</i>
GAD	<i>Grafo Acíclico Dirigido</i>
GCR	<i>Grupo Crítico de Referencia</i>
GCS	<i>Gestión de la Configuración Software</i>
GDPA	<i>Graphical Development Process Assistant</i>
GME	<i>Generic Modeling Environment</i>
GQM	<i>Goal-Question-Metric</i>
HIP	<i>Human Information Processing</i>
HMM	<i>Herramienta de Meta-Modelado</i>
HTML	<i>HyperText Markup Language</i>
I+D	<i>Investigación y Desarrollo</i>
IA-SI	<i>Investigación-Acción en Sistemas de Información</i>
ICEIS	<i>International Conference on Enterprise Information Systems</i>
ICSE	<i>International Conference on Software Engineering</i>
ICSM	<i>International Conference on Software Maintenance</i>
ICSR	<i>International Conference on Software Reuse</i>
IDL	<i>Interface Definition Language</i>
IEC	<i>International Electrotechnical Commission</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IEEE-CS	<i>IEEE – Computer Society</i>
IFIP	<i>International Federation for Information Processing</i>
IPSE	<i>Integrated Project Support Environment</i>
IPSEN	<i>Integrated, Incremental, Interactive Project Support Environment</i>
IRDS	<i>Information Resource Dictionary System</i>
IS	<i>Ingeniería del Software</i>
ISACF	<i>Information Systems Audit and Control Foundation</i>
ISEE	<i>Integrated Software Engineering Environment</i>
ISESS	<i>International Software Engineering Standards Symposium</i>
ISF	<i>Integrated Software Factory</i>
ISO	<i>International Standardization Organization</i>
ISPL	<i>Information Services Procurement Library</i>
IT	<i>Information Technologies</i>
ITM	<i>Information Technology and Management</i>
ITS-CMM	<i>IT Service Capability Maturity Model</i>
IWPC	<i>International Workshop on Program Comprehension</i>
IWPSE	<i>International Workshop on Principles of Software Evolution</i>
IWSM	<i>International Workshop on Software Measurement</i>
JADE	<i>Java Agent DEvelopment framework</i>
JSME	<i>Journal of Software Maintenance and Evolution</i>
JSS	<i>Journal of Systems and Software</i>
KLCF	<i>Kilo-Líneas de Código Fuente</i>
KPA	<i>Key Process Area (Área de Proceso Clave)</i>
LCF	<i>Líneas de Código Fuente</i>
LMP	<i>Lenguaje de Modelado de Procesos</i>
LNCS	<i>Lecture Notes in Computer Science</i>
MDC	<i>Meta-Data Coalition</i>
MOF	<i>Meta-Object Facility</i>
MP	<i>Modelo de Proceso</i>
MSI	<i>Mantenimiento de Sistemas de Información</i>
MT-ECMA	<i>Marco de Trabajo ECMA</i>
MUC	<i>Mantenimiento Correctivo Urgente</i>
MVP-L	<i>Multi View Process Modeling Language</i>
MwR	<i>Maintenance with Reuse</i>
NASA	<i>National Aeronautics and Space Administration</i>
NATO	<i>North Atlantic Treaty Organization</i>
NIST	<i>National Institute of Standards and Technology</i>
OCD	<i>Objetivos de Control Detallados</i>

OCG	Objetivos de Control Generales
OCL	<i>Object Constraint Language</i>
OG	Objetivo General
OIM	<i>Open Information Model</i>
OMG	<i>Object Management Group</i>
OO	Orientado a Objetos
OP	Objetivo Parcial
PCIS	<i>Portable Common Interface Set</i>
PCTE	<i>Portable Common Tool Environment</i>
PF	Puntos Función
PIE	<i>Process Instance Evolution</i>
PIF	<i>Process Interchange Format</i>
PM	Petición de Modificación (o de Mantenimiento)
PMBOK	<i>Project Management Body of Knowledge</i>
PMI	<i>Project Management Institute</i>
PMS	Proceso de Mantenimiento del Software
PPD	Dependencias Producto/Proceso
PPD-EP	Paquete de Experiencia PPD
PPS	Programa del Proceso Software
PROFES	<i>International Conference on Product Focused Software Process Improvement</i>
PROMENADE	<i>PROcess-oriented Modelling and ENActment of software Developments</i>
PS	Proceso Software
PSEE	<i>Process-centered Software Engineering Environment</i>
PSL	<i>Process Specification Language</i>
PT	Producto de Trabajo
PYMES	Pequeñas y Medianas Empresas
Q-MOPP	<i>Qualitative Evaluation of Maintenance Organizations, Processes and Products</i>
RDF	<i>Resource Description Framework</i>
REFSENO	<i>Representation Formalism for Software Engineering Ontologies</i>
RPC	<i>Remote Procedure Call</i>
RUP	Proceso Unificado de desarrollo de software de Rational
SCI	<i>Science Citation Index</i>
SDL	<i>Specification and Description Language</i>
SDS	<i>Schema Definition Sets</i>
SEE	<i>Software Engineering Environment</i>
SEN	<i>ACM SIGSOFT Software Engineering Notes</i>
SGBD	Sistema de Gestión de Bases de Datos
SGC	Sistema de Gestión de Conocimiento
SGFT	Sistema de Gestión de Flujos de Trabajo
SGO	Sistema de Gestión de Objetos
SI	Sistema de Información
SOFT	<i>IEEE Software</i>
SPEM	<i>Software Process Engineering Metamodel Specification</i>
SQA	<i>Software Quality Assurance</i> (Aseguramiento de la Calidad del Software)
SQL	<i>Structured Query Language</i>
STAR	<i>Software Technology for Adaptable, Reliable Systems</i>
SWEBOK	<i>Software Engineering Body of Knowledge</i>
TI	Tecnologías de la Información
TIC	Tecnologías de la Información y Comunicaciones
TPS	Tecnología de Proceso Software
TSE	<i>IEEE Transactions on Software Engineering</i>
UCLM	Universidad de Castilla-La Mancha
UML	<i>Unified Modeling Language</i> (Lenguaje Unificado de Modelado)
USAF	<i>United States Air Force</i>
W3C	<i>World Wide Web Consortium</i>
WfMC	<i>Workflow Management Coalition</i>
WSDL	<i>Web Services Description Language</i>
XMI	<i>XML Metadata Interchange</i>

XML	<i>Extensible Markup Language</i> (Lenguaje de Marcas Extensible)
XP	<i>eXtreme Programming</i>
XPDL	<i>XML Process Definition Language</i>

2. Referencias Bibliográficas.

- ACM (2002a): Computing Classification System (2002 extended version). Disponible en <http://www.computer.org/mc/keywords/ACMtaxonomy.htm>.
- ACM (2002b): Association for Computing Machinery - *ACM Portal to Computing Literature Library*. Disponible en <http://portal.acm.org/portal.cfm>.
- Acuña, S.T. y Ferré, X. (2001): Software Process Modelling. Proceedings of the *International Conference on Information Systems, Analysis and Synthesis (ISAS'2001)*, volume 1. Orlando, Florida (Estados Unidos), pp. 237-242.
- Alloui, I., Beydeda, S., Cîmpan, S., Gruhn, V., Oquendo, F. y Schneider, C. (2000): Advanced Services for Process Evolution: Monitoring and Decision Support. En Conradi, E. (editor); "*Software Process Technology: 7th European Workshop, EWSPT'2000*". LNCS 1780, Springer-Verlag, pp. 21-37.
- Ambriola, V., Ciancarini, P. y Montangero, C. (1990): Software Process Enactment in Oikos. En Taylor, R.N. (editor); Fourth ACM SIGSOFT Symposium on Software Development Environments. ACM Press. Special issue of *Software Engineering Notes*, 15(6), pp. 183-192.
- Ambriola, V., Conradi, R. y Fuggetta, A. (1997): Assessing Process-centered Software Engineering Environments. *ACM Transactions on Software Engineering and Methodology*, 6:3, 1997, pp. 283-328.
- Antoniol, G., Di Penta, M., Casazza, G., Di Lucca, G.A. y Rago, F. (2001): A Queue Theory-Based Approach to Staff Software Maintenance Centers. *IEEE International Conference on Software Maintenance (ICSM'01)*. 7-9 noviembre, Florencia (Italia), pp. 510-519.
- Arbaoui, S. y Oquendo, F. (1994): PEACE: Goal-oriented Logic-based Formalism for Process Modeling. En Finkelstein, A., Kramer, J. y Nuseibeh, B. (editors), *Software Process Modeling and Technology*, Research Studies Press.
- Arbaoui, S., Lonchamp, J., Montangero, C. (1999): The Human Dimension of the Software Process. En "*Software Process: Principles, Methodology and Technology*". LNCS 1500, Springer-Verlag, pp. 165-199.
- Arnold, R.S. (1986): An Introduction to Software Restructuring. *Tutorial on Software Restructuring*. IEEE Computer Society, pp. 1-11.
- Aversano, L., Betti, S., De Lucia, A., y Stefanucci, S. (2001): Introducing Workflow Management in Software Maintenance Processes. *IEEE International Conference on Software Maintenance (ICSM)*. Florencia (Italia). IEEE Computer Society Press, pp. 441-450.
- Avison, D., Lan, F., Myers, M. y Nielsen, A. (1999): Action Research. *Communications of the ACM*, 42(1), 94-97.
- Balzer, R., Cheatham, T.E., y Green, C. (1983): Software Technology in the 1990's: Using a New Paradigm". *IEEE Computer*, 16(11), pp. 39-45.
- Bandinelli, S.C. y Fuggetta, A. (1993): Computational Reflection in Software Process Modeling: The SLANG Approach. *International Conference on Software Engineering (ICSE)*, pp. 144-154.
- Bandinelli, S.C., Fuggetta, A., Ghezzi, C. y Lavazza, L. (1994): "SPADE: An Environment for Software Process Analysis, Design, and Enactment". En "*Software Process Modelling and Technology*". Research Studies Press, pp. 223-247.
- Banker, R.D., Datar, S.M. y Kemerer, C.F. (1991): A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects. *Management Science*, 37(1), pp 1-18.

- Banker, R.D., Datar, S.M., Kemerer, C.F. y Zweig, D. (2002): Software Errors and Software Maintenance Management. *Journal of Information Technology & Management*. Vol. 3, Kluwer Academic Publishers, pp 25-41.
- Basili, V.R. (1990): Viewing Maintenance as Reuse-Oriented Software Development. *IEEE Software*, 7(1), pp. 19-25.
- Basili, V.R. y Rombach, H.D. (1991): Support for Comprehensive Reuse. *IEEE Software Engineering Journal*. 6(5), pp. 303-316.
- Basili, V.R., Caldiera, G. y Rombach, H.D. (1994): Goal Question Metric Paradigm. En Marciniak, J.J. (editor); *Encyclopedia of Software Engineering*, vol. 1. John-Wiley & Sons, pp. 528-532.
- Basili, V., Briand, L., Condon, S., Kim, Y., Melo, W. y Valett, J.D. (1996): Understanding and Predicting the Process of Software Maintenance Releases. Proceedings of the *International Conference on Software Engineering* (ICSE'96). IEEE Computer Society, Los Alamitos, CA (USA), pp. 464-474.
- Basili, V.R. (2000): Using Experiments to Build a Body of Knowledge. *7th European Workshop on Software Process Technology* (EWSPT 2000), keynote slides. Salzburg (Austria), February 22-25.
- Basili, V., Lindvall, M., y Costa, P. (2001a): Implementing the Experience Factory Concepts as a Set of Experience Bases. *13th International Conference on Software Engineering and Knowledge Engineering* (SEKE'99), pp. 102-109.
- Basili, V., Tesorieto, R., Costa, P., Lindvall, M., Rus, I., Shull, F. y Zekowitz, M. (2001b): Building an Experience Base for Software Engineering: A Report on the First CeBASE eWorkshop. *Third Product Focused Software Process Improvement* (PROFES'2001), pp. 110-125.
- Basili, V. y Seaman, C. (2002): The Experience Factory Organization. *IEEE Software*, 19(3), pp 30-31.
- Baskerville, R. y Wood-Harper, A.T. (1996): A Critical Perspective on Action Research as a Method for Information Systems Research. *Journal of Information Technology*, (11) 3, pp. 235-246.
- Baskerville, R. (1997): Distinguishing Action Research From Participative Case Studies. *Journal of Systems and Information Technology*, (1) 1, pp. 25-45.
- Baskerville, R. (1999): Investigating Information Systems with Action Research. *Communications of the Association for Information Systems*. Vol 2, article 19. Disponible en http://cis.gsu.edu/~rbaskerv/CAIS_2_19/index.html (accedido el 16-diciembre-2001).
- Becker-Kornstaedt, U., Hamann, D. y Verlage, M. (1997): *Descriptive Modeling of Software Processes*. IESE-Report 045.97, Fraunhofer, Alemania.
- Becker-Kornstaedt, U. y Webby, R (1999): *A Comprehensive Schema Integrating Software Process Modeling and Software Measurement*. Fraunhofer IESE – Report No 047.99, August 1999. Disponible en http://www.iese.fhg.de/Publications/iese_reports/.
- Belkhatir, N., Estublier, J. y Melo, W.L. (1993): Software Process Model and Work Space Control in the Adele System. Proceedings of the *Second International Conference on the Software Process: Continuous Software Process Improvement*. Berlín (Alemania), 25-26 febrero. IEEE Computer Society, pp. 2-11.
- Benali, K. y Derniame, J.C. (1992): Software Process Modeling: What, Who and When. *2nd European Workshop on Software Process Technology* (EWSPT'92). Trondheim (Noruega), LNCS 635, Springer-Verlag, pp. 21-25.
- Bengtsson, P. y Bosch, J. (1999): Architecture Level Prediction of Software Maintenance. *Third European Conference on Software Maintenance and Reengineering* (CSMR). 3-5 marzo, Amsterdam (Países Bajos), pp. 139-147.
- Bennett, K.H., Martil, R. y Zuylen H.V. (1990): *A Model of Software Reconstruction*. Informe técnico. Centre of Software Maintenance. Durham (Reino Unido).

- Bennet K.H. y Rajlich V.T. (2000): Software Maintenance and Evolution: a Roadmap. En Finkelstein, A. [editor]: The Future of Software Engineering, *International Conference on Software Engineering* (ICSE'2000). Limerick (Irlanda), 4-11 junio, pp. 73-87.
- Bennett, K.H., Munro, M., Gold, N., Layzell, P., Budgen, D. y Brereton, P. (2001): An Architectural Model for Service-Based Software with Ultra Rapid Evolution. *International Conference on Software Maintenance* (ICSM). 6-10 noviembre, Florencia (Italia). IEEE Computer Society, pp. 292-300.
- Ben-Shaul, I.Z. y Kaiser, G.E. (1998): Federating Process-Centered Environments: the Oz Experience. *Automated Software Engineering*. 5(1), Kluwer Academic Publishers, pp. 97-132.
- Bertrand, T. y Bézivin, J. (2000): Ontological Support for Business Process Improvement. En Bustard, D., Kawalek, P. y Norris, M. (editores): *Systems Modeling for Business Process Improvement*. Artech House Publishers, capítulo 20, pp. 313-331.
- Birk, A., Hamann, D. y Pfahl, D. (1998): *Goal-Driven Planning of Maintenance Process*. IESE-Report 011.98, Fraunhofer, Alemania.
- Blázquez, M., Fernández, M., García-Pinar, J.M. y Gómez-Pérez, A. (1998): Building Ontologies at the Knowledge Level using the Ontology Design Environment. 11th *Knowledge Acquisition Workshop* (KAW'98). Banff (Canadá), 18-23 abril.
- Boehm, B.W. (1981): *Software Engineering Economics*. Prentice-Hall.
- Boehm, B.W., Clark, B.K., Horowitz, E., Westland, J.C., Madachy, R. y Selby, R.W. (1995): Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. *Annals of Software Engineering*. vol 1, pp. 57-94.
- Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R. y Reifer, D. (2001): *Software Cost Estimation with COCOMO II*. Prentice-Hall Publishers.
- Breton, E. y Bézivin, J. (2000): An Overview of Industrial Process Meta-Models. 13th *International Conference of Software & System Engineering and their Applications* (ICSSEA'2000). 5-8 diciembre, París (Francia).
- Breton, E. y Bézivin, J. (2001): Model-Driven Process Engineering. 25th *Annual International Computer Software and Applications Conference* (COMPSAC'01). 8-12 octubre, Chicago (Estados Unidos), pp. 225-232.
- Breton, E. y Bézivin, J. (2002): Weaving Definition and Execution Aspects of Process Meta-Models. 35th Annual Hawaii International Conference on System Sciences (HICSS'02). 7-9 enero, Big Island, Hawaii (Estados Unidos), pp. 290b.
- Briand, L.C., El Emam, K. y Bomarius, F. (1998a): COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking and Risk Assessment. *International Conference on Software Engineering*. 19-25 abril, Kyoto (Japón). IEEE Computer Society Press, pp. 340-349.
- Briand, L.C., Kim, Y-M., Melo, W., Seaman, C., y Basili, V.R. (1998b): Q-MOPP: Qualitative Evaluation of Maintenance Organizations, Processes and Products. *Journal of Software Maintenance: Research and Practice*, 10(4), pp. 249-278.
- Briand, L.C., Wüst, J., Daly, J.W. y Porter, V. (2000): Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems. *Journal of Systems and Software*, (51), pp. 245-273.
- Bröckers, A., Lott, C.M., Rombach, H.D. y Verlage, M. (1995): *MVP-L Language Report Version 2*. Fachbereich Informatik TR 265/95, Universität Kaiserslautern (Germany).
- Calero, C., Piattini, M., Genero, M., Serrano, M.A. y Caballero, I. (2000): Metrics for Relational Database Maintainability. Workshop of the *United Kingdom Academy for Information Systems* (UKAIS'2000). Cardiff (reino Unido), 26-28 abril.
- Calzolari, F., Tonella, P. y Antoniol, G. (1998): Modelling Maintenance Effort by Means of Dynamic Systems. *Second Euromicro Conference on Software Maintenance and Reengineering* (CSMR). Los Alamitos, California (Estados Unidos). IEEE Computer Society, pp. 150-156.

- Canós J.H., Penadés M.C., Carsí J.A. (1999): From Software Process to Workflow Process: the Workflow Livecycle. *International Process Technology Workshop (IPTW'1999)*. Villars de Lans (Francia), 1-3 septiembre.
- Canós, J.H. y Penadés, M.C. (2000): Sistemas de Flujo de Trabajo. X Escuela de Verano de Informática. Albacete, 10-12 julio.
- Carr, M. y Wagner, C. (2002): A Study of Reasoning Processes in Software Maintenance Management. *Journal of Information Technology & Management*. Vol. 03, Kluwer Academic Publishers, pp 181-203.
- Castellani, X. (1998): Evaluation of Models Defined with Charts of Concepts: Application to the UML Model. *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'98)*. Pisa (Italia), 8-9 Junio.
- Champlain, J. (1998): *Auditing Information Systems - A Comprehensive Reference Guide*. John Wiley & Sons (Estados Unidos).
- Chan, T. (2000): Beyond Productivity in Software Maintenance: Factors Affecting Lead Time in Servicing Users' Requests. *IEEE International Conference on Software Maintenance (ICSM'00)*. 11-14 octubre, San José, California (Estados Unidos), pp. 228-235.
- Chapin, N. (2002): Software Maintenance and Organizational Health and Fitness. En Polo, M., Piattini, M. y Ruiz, F. (editores); *Advances in Software Maintenance Management: Technologies and Solutions*. Idea Group Publishing (Estados Unidos), capítulo I.
- Chein, I., Cook, S.W. y Harding, J. (1948): The Field of Action Research. *The American Psychologist*, 3, pp. 43-50.
- Chen, J.R., Wolfe, S. y Wragg, S. (2000): A Distributed Multi-Agent System for Collaborative Information Management and Sharing. *ACM CIKM International Conference on Information and Knowledge Management*, McLean, Virginia (USA). 6-11 noviembre, pp. 382-388.
- Chikofsky, E.J. y Cross, J.H. (1990): Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, 7(1), pp. 13-17.
- Chiriatti, K. (1999): Improvement of Maintenance Process using Case Tools. *Euromicro Workshop on Software Process and Product Improvement (PROFES)*. 8-10 septiembre, Milán (Italia).
- CiteSeer (2003): *Citation Index for Computer Science*. Disponible en <http://citeseer.nj.nec.com/>.
- Cockburn, A. (2000): "Selecting a Project's Methodology". *IEEE Software*, 17(4), pp. 64-71.
- Conradi, R., Jaccheri, M.L., Mazzi, C., Nguyen, M.N. y Aarsten, A. (1992): Design, Use and Implementation of SPELL, a Language for Software Process Modelling and Evolution. *European Workshop on Software Process Technology (EWSPT)*. Trondheim (Noruega), pp. 167-177.
- Conradi, R., Fernström, C. y Fuggetta, A. (1994): Concepts for Evolving Software Processes. En Finkelstein, A., Kramer, J. y Nuseibeh, B. (editors), *Software Process Modeling and Technology*, Research Studies Press, pp. 9-31.
- Conradi, R. y Jaccheri, M.L. (1999): Process Modelling Languages. En "Software Process: Principles, Methodology and Technology". LNCS 1500, Springer-Verlag, pp. 27-52.
- COSO (1994): *Internal Control - Integrated Framework*. Committee of Sponsoring Organizations of the Treadway Commission; American Institute of Certified Accountants. New Jersey (Estados Unidos). Disponible en <http://www.coso.org/>.
- Crawley, S., Davis, S., Indulska, J., McBride, S. y Raymond, K. (1997a): Meta Information Management. 2nd IFIP *International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS'97)*. Canterbury (Reino Unido), julio.
- Crawley, S., Davis, S., Indulska, J., McBride, S. y Raymond, K. (1997b): Meta-meta is better-better!. *IFIP International Working Conference on Distributed Applications and Interoperable Systems (DAIS'97)*. Cottbus (Alemania), 12 pgs.

- CSB (2002): Alf-Christian Achilles. *The Collection of Computer Science Bibliographies*. Disponible en <http://liinwww.ira.uka.de/bibliography/index.html>.
- CSI (2000): Consejo Superior de Informática; MÉTRICA versión 3; Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información. Ministerio de Administraciones Públicas. Disponible en <http://www.map.es/csi/metrica3/index.html>.
- CSI (2002): Consejo Superior de Informática; *MAGERIT versión 1.0; Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información de las Administraciones Públicas*. Ministerio de Administraciones Públicas. Disponible en <http://www.map.es/csi/pg5m20.htm>.
- Cugola, G., Di Nito, E., Ghezzi, C., Mantione, M. (1995): How to Deal with Deviations During Process Model Enactment, *Proceedings of the 17th international Conference on Software Engineering*. Seattle, Washington (Estados Unidos), 24-28 abril, pp. 265-273.
- Cugola, G. y Ghezzi, C. (1998): Software Processes: a Retrospective and a Path to the Future. *Software Process: Improvement and Practice*, 4(3), 1998, pp. 101-123.
- Cunin, P.Y. (2000): The PIE Project: An Introduction. En Conradi, E. (editor); “*Software Process Technology: 7th European Workshop, EWSPT’2000*”. LNCS 1780, Springer-Verlag, pp. 1-6.
- Curtis, B., Kellner, M. y Over, J. (1992): Process Modeling. *Communications of ACM*, 31(11), pp. 1268-1287.
- Dami, S., Estublier, J., y Amieur, M. (1998): APEL: a Graphical Yet Executable Formalism for Process Modeling. *Automated Software Engineering*, 5(1), Kluwer Academic Publisher, pp. 60-96.
- DBLP (2002): *DBLP Computer Science Bibliography*. Disponible en <http://dblp.uni-trier.de/>.
- De Vogel, M. (1999): Outsourcing and Metrics. 2nd *European Measurement Conference (FESMA’99)*, pp. 217-225.
- Deiters, W. y Gruhn, V. (1994): The FUNSOFT Net Approach to Software Process Management. *International Journal of Software Engineering and Knowledge Engineering*, 4(2), pp. 229- 256.
- Deridder, D. y Wouters, B. (1999): The Use of Ontologies as a Backbone for Software Engineering Tools. *Fourth Australian Knowledge Acquisition Workshop (AKAW’99)*. 5-6 diciembre, Sydney (Australia).
- Deridder, D. (2002): A Concept-Oriented Approach to Support Software Maintenance and Reuse Activities. *5th Joint Conference on Knowledge-Based Software Engineering (JCKBSE2002)*. 11-13 septiembre, Maribor (Eslovenia).
- Derniame, J.C. et al (1994): Life-Cycle Process Support in PCIS, or It Is Time to Think about Software Process Formalisms Standardization. *Proceedings of the PCTE’94*, Technical Journal No.2, San Francisco, USA, November 1994.
- Derniame, J.C., Kaba, B.A., y Wastell, D. [editores] (1999a): *Software Process: Principles, Methodology and Technology*. LNCS 1500, Springer-Verlag.
- Derniame, J.C., Kaba, B.A., y Warboys, B. (1999b): The Software Process: Modelling and Technology. En “*Software Process: Principles, Methodology and Technology*”. LNCS 1500, Springer-Verlag, pp. 1-13.
- Devedzic, V. (2002): Understanding Ontological Engineering. *Communications of ACM*. 45(4), pp. 136-144.
- Dishaw, M.T., y Strong, D.M. (1998): Assessing Software Maintenance Tool Utilization Using Task-Technology Fit and Fitness-for-use Models. *Journal of Software Maintenance: Research and Practice*, 10(3), pp. 151-179.
- Dolado, J. y Fernández, L. (1999): ¿Merece la pena usar los puntos de función?. *Novática*, vol 140, pp. 57-62.
- Dowson, M., Nejme, B. y Riddle, W. (1991): Fundamental Software Process Concepts. En Ambriola, V., Fugetta, A. y Conradi, R. (editors); *First European Workshop on Software Process Modeling*. AICA Press, pp. 16-37.

- Dowson, M. y Fernström, C. (1994): Towards Requirements for Enactment Mechanisms. *Third European Workshop on Software Process Technology* (EWSPT'94). Villad-de-Lans, France, February 1994. LNCS 772, Springer-Verlag, pp. 90-106.
- Duddy, K. (2002): UML2 must enable a family of languages. *Communications of ACM, special issue: What UML should be*. 45(11), pp. 73-75.
- EC (1996): *Euromethod Project - Euromethod version 1*. European Comission. Disponible en <http://projekte.fast.de/Euromethod/>.
- ECMA (1993): *Reference Model for Frameworks of Software Engineering Environments*, 3rd edition. European Computer Manufacturers Association, TR/55, June 1993. Disponible en <http://www.ecma.ch/ecma1/TECHREP/E-TR-055.HTM>.
- ECMA (1994): *Reference Model for Project Support Environments*. European Computer Manufacturers Association, TR/69, December 1994. Disponible en <http://www.ecma.ch/ecma1/TECHREP/E-TR-069.HTM>.
- ECMA (1997): *Portable Common Tool Environment (PCTE) – Abstract Specification*, 4th edition. European Computer Manufacturers Association, standard 149, December 1997. Disponible en <http://www.ecma.ch/ecma1/STAND/ECMA-149.HTM>.
- Elsevier (2002): Elsevier Science – *Journal of Systems and Software; Search Facilities*. Disponible en <http://www.elsevier.nl/gej-ng/10/29/11/show/Search/>.
- Engels, G., Groenewegen, L. y Dassen, R. (1998): Formalization of the Software Process Modelling Language SOCCA. En Gruhn, V. (editor); *6th European Workshop on Software Process Technology* (EWSPT '98). LNCS 1487, Springer-Verlag, pg. 151.
- Eriksson, H.-E. y Penker, M. (2000): *Business Modelling with UML: Business Patterns at Work*. OMG Press, John Wiley & Sons.
- Estay, C., y Pastor, J. (2000a): Improving Action Research in Information Systems with Project Management. In Chung, M. (2000); *Proceedings of the 2000 Americas Conference on Information Systems*. Long Beach, California. 11-13 August. pp, 1558-1561.
- Estay, C., y Pastor, J. (2000b): Towards a project structure for Action-Research in Information Systems. *10th Annual Business and Information Technology Conference* (BIT'2000). Manchester (United Kingdom), November 1-2.
- Estay, C., y Pastor, J. (2001): Un Modelo de Madurez para Investigación-Acción en Sistemas de Información.. *VI Jornadas de Ingeniería del Software y Bases de Datos* (JISBD'2001). Ciudad Real (España), Noviembre 21-23, pp. 265-281.
- Estublier, J., Cunin, P.Y. y Belkhatir, N. (1998): Architectures for Process Support System Interoperability. *Proceedings of the Fifth International Conference on the Software Process* (ICSP'98), 15-17 Junio, Chicago (Estados Unidos), pp. 137-147.
- Falbo, R.A., Menezes, C.S., Rocha, A.R. (1998): Using Ontologies to Improve Knowledge Integration in Software Engineering Environments. *4th International Conference on Information Systems Analysis and Synthesis* (ISAS'98). Orlando, Florida (Estados Unidos).
- Falbo, R.A., Menezes, C.S., Rocha, A.R. (1999): Using Knowledge Servers to Promote Knowledge Integration in Software Engineering Environments. *11th International Conference on Software Engineering and Knowledge Engineering* (SEKE'99). 17-19 junio, Kaiserslautern (Alemania).
- Feiler, P.H. y Humphrey, W.S. (1993): *Software Process Development and Enactment: Concepts and Definitions*. *Proceedings of the Second International Conference on the Software Process*. IEEE Computer Press, pp. 41-53.
- Fensel, D. (2000): *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer Verlag (Alemania), ISBN 3540416021.
- Fenton, N.E. y Ohlsson, N. (2000): Quantitative Analysis of Faults and Failures in a Complex Software System. *IEEE Transactions on Software Engineering*, 26(8), pp. 797-814.

- Fernández, M., Gómez-Pérez, A., y Juristo, N. (1997): METHONTOLOGY: From Ontological Art Towards Ontological Engineering. *AAAI Spring Symposium*. University of Stanford; Palo Alto, California (Estados Unidos), 24-25 marzo, pp. 33-40.
- Fernström, C. (1993): Process WEAVER: Adding Process Support to UNIX. *Proceedings of the Second International Conference on the Software Process*. IEEE Computer Society Press, pp. 12-26.
- Finkelstein, L. (1984): A review of the fundamental concepts of measurement. *Measurement*, 2(1), pp. 25-34.
- Fioravanti, F., Nesi, P. y Stortoni, F. (1999): Metrics for Controlling Effort During Adaptive Maintenance of Object-Oriented Systems. *International Conference on Software Maintenance (ICSM'99)*, 30 agosto-3 septiembre, Oxford (Reino Unido), pp. 483-492.
- Fioravanti, F. (2002): The Impact of extreme Programming on Maintenance. En Polo, M., Piattini, M. y Ruiz, F. (editores); *Advances in Software Maintenance Management: Technologies and Solutions*. Idea Group Publishing (Estados Unidos), capítulo III.
- FIPA (2000): *FIPA Agent Software Integration Specification*. Foundation for Intelligent Physical Agents. Disponible en <http://www.fipa.org/specs/fipa00079/>.
- FIPA (2002): *FIPA ACL Message Structure Specification*. Disponible en <http://www.fipa.org/specs/fipa00061/>.
- Franch, X. y Ribó, J.M. (1999a): Using UML for Modelling the Static Part of a Software Process. En “UML'99 - The Unified Modeling Language”. LNCS 1723, Springer-Verlag, pp. 292 - 307.
- Franch, X. y Ribó, J.M. (1999b): Some Reflexions in the Modelling of Software Processes. *International Process Technology Workshop (IPTW)*. 1-3 septiembre, Villars de Lans (Francia).
- French, W.L. y Bell, C.H. (1996): *Organizational Development: Behavioral Science Interventions for Organization Improvement*. Prentice Hall: London.
- Fuggetta, A., Godart, C. y Jahnke, J. (1999): Architectural Views and Alternatives. En “Software Process: Principles, Methodology and Technology”. LNCS 1500, Springer-Verlag, pp. 95-116.
- Fuggetta, A. (2000): Software Process: A Roadmap. 22nd *International Conference on Software Engineering (ICSE'2000)*, Future of Software Engineering Track, June 4-11, Limerick (Irlanda). ACM.
- García, F. (2001): *Arquitectura Software Abierta para Repositorios de Datos y Metadatos*. Universidad de Castilla-La Mancha, Dep. de Informática. Proyecto fin de carrera, septiembre 2001.
- García, F., Márquez, L., Ruiz, F., Piattini, M. y Polo, M. (2001a): A Tool for the Management of the Software Maintenance Process. En “Advances in Signal Processing and Computer Technologies”. WSES Press (Estados Unidos), ISBN 960-8052-37-8, pp. 228-232.
- García, F., Ruiz, F., Piattini, M., Márquez, L. y Polo, M. (2001b): Propuesta de Repositorio basado en XMI para Metamodelado de Procesos Software. *XXVII Conferencia Latinoamericana de Informática (CLEI'01)*. Mérida (Venezuela), 24-28 septiembre, pp.70.
- García, F., Ruiz, F., Piattini, M. y Polo, M. (2002a): Conceptual Architecture for the Assessment and Improvement of Software Maintenance. En “Enterprise Information Systems IV”. Kluwer Academic Publishers (Países Bajos), pp. 219-226. ISBN 1-4020-1086-9.
- García, F., Ruiz, F., Piattini, M. y Polo, M. (2002b): Conceptual Architecture for the Assessment and Improvement of Software Maintenance. *4th International Conference on Enterprise Information Systems (ICEIS'02)*. 2-6 abril, Ciudad Real (España), pp. 610-617.
- García, F., Ruiz, F. y Piattini, M. (2003a): Metamodelado y Medición para la Evaluación y Mejora del Proceso de Mantenimiento del Software. *Computación y Sistemas*, IPN-México (pendiente de evaluación).
- García, F., Ruiz, F., Cruz, J.A. y Piattini, M. (2003b): Integración del Metamodelado y la Medición para la Mejora de los Procesos Software. *6º Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software (IDEAS'2003)*. Asunción (Paraguay), 30-abril/2-mayo.

- Garg, P.K. y Jazayeri, M. (1996): Process-centered Software Engineering Environments: A Grand Tour. En Fuggetta, A. y Wolf, A. (editores); *Software Process*. John Wiley & Sons.
- GDPA (2002): *Graphical Development Process Assitant*. Universidad de Bremen (Alemania). Disponible en <http://www.v-modell.de/home.htm>.
- Género, M., Ruiz, F., Piattini, M., Calero, C. y García, F. (2003): An Ontology for Software Measurement. *International Conference on Software Engineering and Knowledge Engineering (SEKE'2003)*. San Francisco, CA (USA), julio 2003 (aceptado).
- Glass, R.L. (1998): Maintenance: Less is not More. *IEEE Software*; 15(4), pp. 67-68.
- Gnyawali, D.R., Stewart, A.C. y Grant J.H. (1997): Creating and Utilization of Organizational Knowledge: An Empirical Study of the Roles of Organizational Learning on Strategic Decision Making. *Academy of Management, Best Paper Proceedings*, pp. 16-20.
- Godart, C. (1999): Cooperation Control in PSEE. En “*Software Process: Principles, Methodology and Technology*”. LNCS 1500, Springer-Verlag, pp. 117-164.
- Gómez-Pérez A (1998): Knowledge Sharing and Reuse. En Liebowitz, J. (editor): *The Handbook of Applied Expert Systems*. CRC Press.
- Graham, I., Henderson-Sellers, B. y Younessi, H. (1997): *The OPEN Process Specification*. ACM Press y Addison-Wesley.
- Gruber, T. (1995): Towards Principles for the Design of Ontologies used for Knowledge Sharing. *International Journal of Human-Computer Studies*, 43(5/6), pp 907-928.
- Gruhn, V. (1994): Communication Support in a Process-centered Software Engineering Environment. *9th International Software Process Workshop*. Airlie, Virginia (Estados Unidos).
- Grundy, J.C. y Hosking, J.G. (1998): “Serendipity: Integrated Environment Support for Process Modelling, Enactment and Work Coordination”. *Automated Software Engineering: Special Issue on Process Technology*, 5(1) , Kluwer Academic Publishers, pp. 27-60.
- Gruninger, M. y Lee, J. (2002): Ontology Applications and Design. *Communications of the ACM*. 45(2), February 2002, pp. 39-41.
- Guarino, N. (1998): Formal Ontology and Information Systems. *Formal Ontology in Information Systems (FOIS'98)*. Trento (Italia), 6-8 junio. IOS Press, pp. 3-15.
- Guizzardi, G., Herre, H. y Wagner, G. (2002): On the General Ontological Foundations of Conceptual Modeling. 21st *International Conference on Conceptual Modeling (ER'2002)*. Tampere (Finlandia), 7-11 octubre.
- Gupta, A., y Govindarajan, V. (2000): Knowledge Flows within Multinational Corporations. *Strategic Management Journal*, 21(4), pp. 473-496.
- Hall, T., Rainer, A., Baddoo, N. y Beecham, S. (2001): An Empirical Study of Maintenance Issues within Process Improvement Programmes in the Software Industry. *IEEE International Conference on Software Maintenance (ICSM'01)*. 7-9 noviembre, Florencia (Italia), pp. 422-430.
- Hanrahan, R., Daud, C., Meiser, K. y Peterson, J. (1994): *Software Engineering Environment Technology Report*. USAF, Software Technology Support Center.
- Hanus, M. (1991): The ALF System: An Efficient Implementation of a Functional Logic Language. En Boley, H. y Richter, M.M. (editores); *International Workshop on Processing Declarative Knowledge (PDK'91)*. LNCS 567, Springer-Verlag, pp. 414-416.
- Heimbigner, D. (1993): A Revisionist Approach to Process Change. *Proceedings of the Eight International Software Process Workshop (ISPW)*. Schloss Dagstuhl (Alemania), 2-4 marzo, pp. 95-97.
- Hikita, T. y Matsumoto, M.J. (2001): Business Process Modeling Based on the Ontology and First-Order Logic. 3rd *International Conference on Enterprise Information Systems (ICEIS'2001)*. Setúbal (Portugal), 7-10 julio, pp. 717-723.
- ISO/IEC (1995): *12207: Information Technology - Software Life Cycle Processes*.

- Hoare, C.A.R. (1985): *Communicating Sequential Processes*. Prentice-Hall.
- Horowitz, E. y Sartaj, S. (1978): *Fundamentals of Computer Algorithms*. Computer Software Engineering Series, Pitman (Reino Unido).
- Humphrey, W.S. (1990): *Managing the Software Process*. Addison-Wesley, Londres (Reino Unido).
- IEE (2003): INSPEC - Bibliographic References for Physics, Computing and Engineering Research. Institution of Electrical Engineers. Disponible en <http://edina.ac.uk/inspec/>.
- IEEE (1993): *IEEE std 1219 - Standard for Software Maintenance*. IEEE Computer Society Press, Estados Unidos.
- IEEE (1995): *STD 1074-1995: IEEE Standard for Developing Software Life Cycle Processes*.
- IEEE-CS (2002): Institute for Electrical and Electronic Engineering - *IEEE Computer Society Digital Library*. Disponible en <http://www.computer.org/publications/dlib/>.
- IFIP (1998): *FRISCO. A Framework of Information Systems Concepts*. International Federation for Information Processing (IFIP), WG 8.1 Task Group, version 1998. Disponible en <ftp://ftp.leidenuniv.nl/pub/rul/fri-full.zip>.
- ISACF (1998): *CobiT: Governance, Control and Audit for Information and Related Technology*, 2nd edition. Information Systems Audit and Control Foundation (Estados Unidos).
- ISI (2003): *ISI Science Citation Index*. Información disponible en <http://www.isinet.com/isi/products/citation/sci/>.
- ISO (2000): *9001-2000: Quality management systems - Requirements*, Dec-2000.
- ISO/IEC (1995): *12207: Information Technology - Software Life Cycle Processes*.
- ISO/IEC (1998a): *15271: Information Technology - Guide for ISO/IEC 12207 (Software Life Cycle Processes)*.
- ISO/IEC (1998b): *15504-1: Information Technology - Software Process Assessment - Part 1: Concepts and Introductory Guide*.
- ISO/IEC (1998c): *15504-2: Information Technology - Software Process Assessment - Part 2: A Reference Model for Processes and Process Capability*.
- ISO/IEC (1998d): *15504-9: Information Technology - Software Process Assessment - Part 9: Vocabulary*.
- ISO/IEC (1998e): *FDIS 14764: Software Engineering - Software Maintenance (draft)*, Dec-1998.
- ISO/IEC (1998f): *15504-3: Information Technology - Software Process Assessment - Part 3: Performing An Assessment*.
- ISO/IEC (1999): *15504-5: Information Technology - Software Process Assessment - Part 5: An Assessment Model and Indicator Guidance*.
- ISO/IEC (2000a): *Framework for ISO/IEC System and Software Engineering Standards*, draft 6.1, July 2000.
- ISO/IEC (2000b): *Amendment to ISO/IEC 12207:1995 - Information Technology - Software Life Cycle Processes*.
- ISO/IEC (2000c): *FDIS 15474-1: Information Technology - CDIF Framework - Part 1: Overview*.
- ISO/IEC (2001a): *Amendment to ISO/IEC TR 15504-2 - Information Technology - Software Process Assessment - Reference Model Extensions For Acquirer Processes*.
- ISO/IEC (2001b): *CD 15940: Information Technology -Software Engineering Environment Services*, working draft 5, May-2001.
- ISO/IEC (2001c): *9126-1: Software Engineering - Product quality - Part 1: Quality model*.
- ISO/IEC (2002): *FDIS 15939: Software Engineering - Software Measurement Process (draft)*, Jan-2002.

- Jaccheri, M.L. y Stålhane, T. (2001): Evaluation of the E3 Process Modelling Language and Tool for the Purpose of Model Creation. En Bomarius, F. y Komi-Sirviö, S. (editores); *Third International Conference on Product Focused Software Process Improvement (PROFES)*. LNCS 2188, Springer-Verlag, pp. 271-281.
- Jäger, D., Schleicher, A. y Westfechtel, B. (1999): Using UML for Software Process Modeling. En Nierstrasz, O. y Lemoine, M. (editores), *7th European Software Engineering Conference (ESEC/FSE '99)*. *ACM Software Engineering Notes*, 21(6), pp. 91-108.
- Jiménez, M.M. (2001): *Herramienta para la Evaluación del Proceso de Mantenimiento*. Universidad de Castilla-La Mancha, Dep. de Informática. Proyecto fin de carrera, septiembre 2001.
- Jorgensen, M. (1995): Experience with the Accuracy of Software Maintenance Task Effort Prediction Models. *IEEE Transactions on Software Engineering*. 21(8), pp. 674-681.
- Junkermann, G., Peuschel, B., Schäfer, W. y Wolf, S. (1994): MERLIN: Supporting Cooperation in Software Development through a Knowledge-based Environment. En “*Software Process Modelling and Technology*”. Research Studies Press, pp. 103-129.
- Kaiser, G.E. (1993): MARVEL 3: A Multi-User Software Development Environment. *Proceedings of the International Logic Programming Symposium*. 26-29 octubre, Vancouver (Canada). MIT Press, pp. 36-39.
- Kajko-Mattsson, M. (1998): A Conceptual Model of Software Maintenance. *20th International Conference on Software Engineering (ICSE)*. 19-25 abril, Kyoto (Japón), pp. 422-425.
- Kajko-Mattsson, M. (1999): Common Concept Apparatus within Corrective Software Maintenance. *IEEE International Conference on Software Maintenance (ICSM'99)*. 30-agosto a 3-septiembre, Oxford (Reino Unido), pp. 287-296.
- Kajko-Mattsson, M. (2001): Towards a Business Maintenance Model. *IEEE International Conference on Software Maintenance (ICSM)*. 6-10 noviembre, Florencia (Italia), pp. 500-509.
- Kajko-Mattsson, M., Westblom, U., Forssander, S., Andersson, G., Medin, M., Ebarasi, S., Fahlgren, T., Johansson, S.E., Törnquist, S. y Holmgren, M. (2001a): Taxonomy of Problem Management Activities. *Fifth European Conference on Software Maintenance and Reengineering (CSMR)*. 14-16 marzo, Lisboa (Portugal), pp. 1-10.
- Kajko-Mattsson, M., Forssander, S. y Olsson, U. (2001b): Corrective Maintenance Maturity Model (CM3): Maintainer's Education and Training. *23rd International Conference on Software Engineering (ICSE)*. 12-19 mayo, Toronto (Canadá). IEEE Computer Society, pp. 610-619.
- Karrer, A., y Penedo, M. (1990): A Survey of Alternative SEE Architectural Approaches. Arcadia project Technical Report 90-004.
- Katayama, T. (1989): A Hierarchical and Functional Software Process Description and its Enaction. *Proceedings of the 11th International Conference on Software Engineering (ICSE)*, pp. 343-352.
- Kelly, S., Lyytinen, K. y Rossi, M. (1996): MetaEdit+ - A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. En Seltheit, A.H., y Farshchian, B.A. (editores), *8th International Conference on Advanced Information Systems Engineering (CAISE'96)*. Heraklion (Grecia). LNCS 1080, Springer-Verlag, pp. 1-21.
- Kempkens, R., Rösch, P., Scott, L. y Zettel, J. (2000): A Multi-Layer Multi-View Architecture for Software Engineering Environments. *Information and Software Technology*, vol 42(2), pp. 141-149.
- Kiepuszewski, B., Hofstede, A.H.M. y Bussler, C. (2000): On Structured Workflow Modelling. En Wangler, B. y Bergman, L. [editores]; *Twelfth International Conference on Advanced Information Systems Engineering (CAiSE'2000)*. Estocolmo (Suecia). LNCS 1789, Springer Verlag, pp. 431-445.
- Kim, H.M. (1999): *Representing and Reasoning about Quality using Enterprise Models*. Ph.D. Thesis. Department of Mechanical and Industrial Engineering, University of Toronto (Canada).

- Kitchenham, B.A., Pfleeger, S.L. y Fenton, N.E. (1995): Towards a Framework for Software Measurement Validation. *IEEE Transactions on Software Engineering*. 21(12), pp. 929-944.
- Kitchenham, B.A. (1996): *DESMET: A Method for Evaluating Software engineering Methods and Tools*. TR96-09, Dep. of Computer Science, Univ. of Keele (UK). August 1996.
- Kitchenham, B.A., Travassos, G.H., Mayrhauser, A., Niessink, F., Schneidewind, N.F., Singer, J., Takada, S., Vehvilainen, R. y Yang, H. (1999): Towards an Ontology of Software Maintenance. *Journal of Software Maintenance: Research and Practice*. 11(6), pp. 365-389.
- Kitchenham, B.A., Hughes, R.T. y Linkman, S.G. (2001a): Modeling Software Measurement Data. *IEEE Transactions on Software Engineering*. 27(9), pp. 788-804.
- Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El-Emam, K. y Rosenberg, J. (2001b): *Preliminary Guidelines for Empirical Research in Software Engineering*. NRC 44158, National Research Council of Canada – Institute for Information Technology, enero 001.
- Klein, M. (2001): Combining and Relating Ontologies: An Analysis of Problems and Solutions. *Workshop on Ontologies and Information Sharing* (dentro del IJCAI'01). Seattle (Estados Unidos), 4-5 agosto.
- Kluwer (2002): *Kluwer Online Gateway - Search Articles - Advanced Search*. Disponible en <http://www.kluweronline.com/>.
- Kobialka, H.U. (1998): Implementing Support for Software Processes in a Process-centered Software Engineering Environment. Tesis doctoral. GMD Research Series No. 15/1998 (Alemania). Disponible en <http://www.gmd.de/publications/research/1998/015/Text.ps>.
- Koch, G.R. (1993): Process Assessment: The 'BOOTSTRAP' Approach. *Information and Software Technology*, 35(6/7), pp. 387-403.
- Kock, N. y Lau, F. (2001): Information Systems Action Research: Serving Two Demanding Masters. *Information Technology & People* (special issue on Action Research in Information Systems), 14(1), pp. 6-11.
- Komi-Sirviö, S., Mäntyniemi, A., y Seppänen, V., (2002): Towards a Practical Solution for Capturing Knowledge for Software Projects. *IEEE Software*. Vol. 19, nº3, pp. 60-62.
- Kontio, J. (2001): *Software Engineering Risk Management: A Method, Improvement Framework, and Empirical Evaluation*. Helsinki University of Technology (Finlandia), Department of Computer Science and Engineering, tesis doctoral. Disponible en <http://lib.hut.fi/Diss/2001/isbn951225655X/>.
- Koskinen, M. (1999): A Metamodelling Approach to Process Concept Customisation and Enactability in MetaCASE. *University of Jyväskylä* (Finlandia), Department of Computer Science and Information Systems; tesis doctoral, febrero 1999.
- Kown, O-C., Shin, G-S., Boldyreff, C. y Munro, M. (1999): Maintenance with Reuse: An Integrated Approach Based on Software Configuration Management. 6th *Asia-Pacific Software Engineering Conference* (APSEC). 7-10 diciembre, Takamatsu (Japón).
- Kung, H.J. y Hsu, C. (1998): Software Maintenance Life Cycle Model. *International Conference on Software Maintenance*. Los Alamitos, California (Estados Unidos). IEEE Computer Society Press, pp. 113-121.
- Lam, W., Loomes, M. y Shankaraman, V. (1999): Managing Requirements Change using Metrics and Action Planning. Third *European Conference on Software Maintenance and Reengineering* (CSMR). 3-5 marzo, Amsterdam (Países Bajos), pp. 122-128.
- Larson, L., Nidiffer, K.E., Rose, L.C., Small, R. y Stankosky, M. (2001): Knowledge Management: Insights from the Trenches. *IEEE Software*, 18(6), pp. 66-68.
- Lau, F. (1997): A Review on the Use of Action Research in Information Systems Studies. En Lee, A.S., Liebenau, J. y Degross, J.I. (1997): *Information Systems Research: Information Systems and Qualitative Research*. Chapman & Hill. London, pp. 31-68.

- Lédeczi, A., Bakay, A., Maróti, M., Völgyesi, P., Nordstrom, G.; Sprinkle, J. y Karsai, G. (2001): Composing Domain-Specific Design Environments. *IEEE Computer* 34(11), pp. 44-51.
- Lee, J. Gruninger, M., Jin, Y., Malone, T., Tate, A. y Yost, G. (1998): The PIF Process Interchange Format and Framework. *Knowledge Engineering Review*, 13(1), pp.91-120.
- Lehman, M.M (1980): Programs, Life Cycles, and Laws of Software Evolution. *Proceedings of the IEEE*, 68(9), pp.1060-1076.
- Lehman, M.M., Perry, D.E. y Ramil, J.F. (1998): Implications of Evolution Metrics on Software Maintenance. *International Conference on Software Maintenance (ICSM)*, IEEE Computer Society Press, pp. 208-217.
- Liu, C., Lin, X., Zhou, X. y Orlowska, M. (1999): Building a Repository for Workflow Systems. *Proceedings of the 31st International Conference on Technology of Object-Oriented Language and Systems*. IEEE Computer Society Press, pp. 348-357.
- Lonchamp, J. (1994): An Assesment Exercise. En Finkelstein, A., Kramer, J. y Nuseibeh, B. (editors); *Software Process Modelling and Technology*. Research Studies Press, Wiley and Sons, pp. 335-356.
- Long, F. y Morris, E. (1993): *An Overview of PCTE: A Basis for Portable Common Tool Environment*. Technical Report CMU/SEI-93-TR-1, March 1993.
- Madhavji, N.H. (1991): The Process Cycle. *Software Engineering Journal*, 6(5), pp. 234-242, 1991.
- Malone, T.W. y Crowston, K. (1994): The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26(1), pp. 87-119.
- Márquez, L. (2001): *Herramienta para Metamodelización de Procesos Software*. Universidad de Castilla-La Mancha, Dep. de Informática. Proyecto fin de carrera, septiembre 2001.
- Márquez, L., García, F., Ruiz, F., Piattini, M. y Polo, M. (2001a): Managing Complexity of Software Processes. *International Symposium on Information Systems and Engineering (ISE'2001)*. Las Vegas (USA), pp. 180-186.
- Márquez, L., García, F., Ruiz, F., Piattini, M. y Polo, M. (2001b): Herramienta para la Representación de Procesos Software mediante Modelos MOF. VII *Congreso Internacional de Ingeniería Informática (ICIE'2001)*. Buenos Aires (Argentina), abril.
- Márquez, L., García, F., Ruiz, F., Piattini, M. y Polo, M. (2001c): Herramienta para Metamodelización de Procesos Software. *Jornadas Chilenas de Computación (ECC'01)*. Punta Arenas (Chile), noviembre. CD-ROM.
- Martínez, A., Canós, J.H. y García-Consuegra, J.D. (2001a): Un Estilo Arquitectónico para la Federación de Sistemas Basados en Procesos. *VI Jornadas de Ingeniería del Software y Bases de Datos*. Almagro (España), pp. 99-111.
- Martínez, A., Canós, J.H. y García-Consuegra, J.D. (2001b): *Estudio del Estado del Arte en Modelado y Ejecución de Procesos Interorganizacionales*. Universidad de Castilla-La Mancha, Departamento de Informática, informe técnico #DIAB-01-06-21.
- Mateos, M.D. (2001): *CREM: Herramienta para la Estimación de Recursos en una Cartera de Proyectos de Mantenimiento*. Universidad de Castilla-La Mancha, Dep. de Informática. Proyecto fin de carrera, diciembre 2001.
- Maurer, F. y Dellen, B. (1998): An Internet based Software Process Management Environment. *ICSE'98 Workshop on Software Engineering over the Internet*. (dentro de ICSE'98). Disponible en <http://sern.cpsc.ucalgary.ca/maurer/ICSE98WS/Submissions/Maurer/ICSE.html>.
- McChesney, I.R. (1995): Toward a Classification Scheme for Software Process Modelling Approaches. *Information and Software Technology*, 37(7), pp. 363-374.
- McGuinness, D.L., Fikes, R., Rice, J. y Wilder, S. (2000): An Environment for Merging and Testing Large Ontologies. En Cohn, A.G., Giunchiglia, F. y Selman, B. [editores]; *Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, pp. 483-493.

- McLeod, R. jr (1990): *Management Information Systems*. McMillan Publishing, New York.
- McNiff, J. (1988): *Action Research. Principles and Practice*, London (UK): Macmillan.
- McTaggart, R. (1991): Principles of Participatory Action Research. *Adult Education Quarterly*, 41(3).
- MDC (1999): *Open Information Model*, v.1.0. Meta Data Coalition, agosto 1999.
- Milner, R. (1989): *Communication and Concurrency*. Prentice-Hall.
- Moore, J.W. (1998): *Software Engineering Standards: A User's Road Map*. IEEE Computer Society Press.
- Munck, R., Oberndorf, P., Ploedereder, E. y Thall., R. (1989): An Overview of DOD-STD-1838A (proposed), the Common APSE Interface Set, revision A. *ACM SIGPLAN Notices*, 24(2), pp. 235-247.
- Munson, J.C. y Khoshgoftaar, T.M. (1992): The Detection of Fault-Prone Programs. *IEEE Transactions on Software Engineering*, 18(5), pp. 423-433.
- Murer, T., Würtz, A., y Scherer, D. (1996): A 3D Model for a Common Understanding of the Software Process. Proceedings of *Asia-Pacific Conference on Computer Human Interaction (APCHI 96)*. Singapur, pp. 318-323.
- Myers, M.D. (1997): Qualitative Research in Information Systems. *MIS Quarterly*, 21(2): 241-242.
- Mylopoulos, J. (1998): Information Modeling in the Time of the Revolution. *Information Systems*, 23(3-4), pp. 127-155.
- Nagl, M. [editor] (1996): *Building Tightly-Integrated Software Development Environments: The IPSEN Approach*. LNCS 1170, Springer-Verlag.
- NATO (1993): *PCIS Architecture: Framework Abstract Specification*, Version 1.0. Tri-Service Group on Communications and Electronics, Special Working Group on Ada Programming Support Environments.
- Nebus, J. (2001): Framing the Knowledge Search Problem: Whom Do We Contact, and Why Do We Contact Them?. *Academy of Management Best Papers Proceedings*, pp h1- h7.
- NEC (2002): NEC ResearchIndex CiteSeer Computer Science - *Citeseer Scientific Literature Digital Library*. Disponible en <http://citeseer.nj.nec.com/cs>.
- Neitzel, A.C. (1999): Managing Risk Management. Crosstalk, The Journal of *Defense Software Engineering*, July, pp. 17-21.
- Nguyen, M.N. y Conradi, R. (1994): Classification of Meta-processes and their Models. *International Conference on Software Process*. Reston-Virginia (Estados Unidos), 10-11 octubre. IEEE Computer Society Press.
- Niessink, F. y Van Vliet, H. (1997): Predicting Maintenance Effort with Function Points. *International Conference on Software Maintenance (ICSM)*. Los Alamitos, California (Estados Unidos). IEEE Computer Society, pp. 32-39.
- Niessink, F. y Van Vliet, H. (1998): Towards Mature Measurement Programs. Second *Euromicro Conference on Software Maintenance and Reengineering (CSMR)*. 8-11 marzo, Florencia (Italia). IEEE Computer Society, pp. 82-88.
- Niessink, F. y Van Vliet, H. (1999a): Measurements Should Generate Value, rather than Data. *Sixth IEEE International Symposium on Software Metrics (METRICS'99)*. IEEE Computer Society Press, Boca Raton (USA), pp. 31-38.
- Niessink, F. y Van Vliet, H. (1999b). *The Vrije Universiteit IT Service Capability Maturity Model*. Technical Report IR-463. Release L2-1.0. Faculty of Sciences, Division of Mathematics and Computer Science. Vrije Universiteit, Amsterdam (Países Bajos).
- Niessink, F. (2000): *Perspectives on Improving Software Maintenance*. Tesis doctoral. Division of Mathematics and Computer Sciences, Faculty of Sciences, Vrije Universiteit. Amsterdam (Países Bajos), marzo 2000. Disponible en <http://www.serc.nl/people/niessink/publications.html>.

- Niessink, F. y Van Vliet, H. (2000): Software Maintenance from a Service Perspective. *Journal of Software Maintenance: Research and Practice*. 12(2), March/April 2000, pp. 103-120.
- Niessink, F. (2001): Perspectives on Improving Software Maintenance. *International Conference on Software Maintenance (ICSM 2001)*. 6-10 noviembre, Florencia (Italia). IEEE Computer Society, pp. 553-556.
- Nuseibeh, B., Kramer, J. y Finkelstein (1993): Expressing the Relationship between Multiple Views in Requirements Specification. 15th *International Conference on Software Engineering (ICSE'93)*.
- O'Neal, J.S., y Carver, D.L. (2001): Analyzing the Impact of Changing Requirements. *IEEE International Conference on Software Maintenance (ICSM'01)*. 7-9 noviembre, Florencia (Italia), pp. 190-195.
- O'Neill, D. (1997): Software Maintenance and Global Competitiveness. *Journal of Software Maintenance: Research and Practice*. 9(6), pp. 379-399.
- Ocampo, C. y Botella, P. (1998): *Some Reflections on Applying Workflow Technology to Software Processes*. Universitat Politècnica de Catalunya, Departamento de Lenguajes y Sistemas Informáticos, informe técnico TR-LSI-98-5-R, Barcelona, 1998.
- O'Dell, C. y Grayson, C.J. (1998): If Only We Knew What We Know: Identification and Transfer on Internal Best Practice. *California Management Review*, 40(3), pp. 154-174.
- Oliveira, K., Rocha, A.R., Travassos, G. y Matwin, S. (1998): Towards Domain-Oriented Software Development Environment for Cardiology. *Conference on Advanced Information Systems Engineering (CAISE'98)*. Pisa (Italia), 8-9 junio.
- Olsina, L.A., Bertoa, M.F., Lafuente, G.J., Martín, M.A., Katrib, M., Vallecillo, A. (2002): Un Marco Conceptual para la Definición y Explotación de Métricas de Calidad. *VII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2002)*. El Escorial (Madrid), 18-21 noviembre.
- OMG (2000a): *Workflow Management Facility Specification*; version 1.2, abril 2000. Object Management Group. Disponible en <http://www.omg.org/technology/documents/formal/mof.htm>.
- OMG (2000b): *XML Metadata Interchange (XMI), version 1.1*. Object Management Group, noviembre 2000.
- OMG (2001a): *OMG Unified Modeling Language Specification*; versión 1.4, septiembre 2001. Object Management Group. Disponible en <http://www.omg.org/technology/documents/formal/uml.htm>.
- OMG (2001b): *Software Process Engineering Metamodel Specification*; adopted specification, diciembre 2001. Object Management Group.
- OMG (2002a): *OMG XML Metadata Interchange (XMI) Specification*; versión 1.2, enero 2002. Object Management Group. Disponible en <http://www.omg.org/technology/documents/formal/xmi.htm>.
- OMG (2002b): *Meta Object Facility (MOF) Specification*; versión 1.4, abril 2002. Object Management Group. Disponible en <http://www.omg.org/technology/documents/formal/mof.htm>.
- OMG (2002c): *The Common Object Request Broker: Architecture and Specification*; versión 3.0, julio 2002. Object Management Group. Disponible en http://www.omg.org/technology/documents/formal/corba_iiop.htm.
- OMG (2002d): *Software Process Engineering Metamodel Specification*; adopted specification, mayo 2002. Object Management Group. Disponible en <http://cgi.omg.org/cgi-bin/doc?ptc/02-05-03>.
- Ossher, H., Harrison, W., y Tarr, P. (2000): Software Engineering Tools and Environments: a Roadmap. *International Conference on Software Engineering (ICSE) - Future of SE Track*. Limerick (Ireland), pp. 261-277.
- Ostolaza, E., Quintano, N. y Satriani, G. (2002): PATTERNS, un Enfoque Práctico a la Gestión del Conocimiento en el Área de Gestión de Proceso Software. *VII Jornadas sobre Innovación y Calidad del Software*, Palma de Mallorca, 11-12 julio.
- Padak, N. y Padak, G. (1994): Guidelines for Planning Action Research Projects. *Ohio Literacy Resource Center*. Disponible en <http://archon.educ.kent.edu/Oasis/Pubs/0200-08.html> (accedido el 20-marzo-2002).

- Paniagua, E., Palma, J.T. y Martín F. (2001): Los Sistemas Multiagente para el Modelado de la Actuación en Organizaciones Humanas. *Revista Iberoamericana de Inteligencia Artificial*. nº14, pp.78-90.
- Parasuraman, A., Zeithaml, V.A., y Berry, L.L. (1985): A Conceptual Model of Service Quality and its Implication for Future Research. *Journal of Marketing*, 49(fall), pp. 41-50.
- Paulk, M.C., Curtis, B., Chrissis, M.B. y Weber, C.V. (1985): The Capability Maturity Model. En Thayer (ed.) (1985). *Software Engineering Project Management*. IEEE Computer Society. pp. 48-59.
- Paulk, M.C., Curtis, B., Chrissis, M.B. y Weber, C.V. (1993): Capability Maturity Model for Software, versión 1.1. Software Engineering Institute, CMU/SEI-93-TR-24.
- Peña, E. (1998): Objetivos de Control y Estructura de CobiT. *I Jornadas de Auditoría Informática (JAI'98)*. 18-19 noviembre, Ciudad Real (España), pp. 93-118.
- Penadés, M.C., Canós, J.H., y Carsí, J.A. (1999a): Hacia una Herramienta de Soporte al Proceso Software Basada en la Tecnología de Workflow. *IV Jornadas de Ingeniería del Software y Bases de Datos*. Cáceres (España), 24-26 noviembre.
- Penadés, M.C., Canós, J.H., y Carsí, J.A. (1999b): From Software Process to Workflow Process: the Workflow Lifecycle. *International Process Technology Workshop (IPTW'99)*. Grenoble (Francia), 13-15 septiembre.
- Perry, B., Taylor, M. y Unruh, A. (1999): Information Aggregation and Agent Interaction Patterns in InfoSleuth. *International Conference on Cooperative Information Systems (CoopIS'99)*. United Kingdom.
- Peuschel, B. y Schäfer, W. (1992): Concepts and Implementation of a Rule-based Process Engine. *Proceedings of the 14th International Conference on Software Engineering (ICSE)*. 11-15 Mayo, Melbourne (Australia), pp. 262-279.
- Pfleeger, S. (1994-1995): Design and Analysis in Software Engineering, Parts 1-5. *Software Engineering Notes*, 19(4), 20(1), 20(2), 20(3) y 20(5).
- Piattini, M. y del Peso, E. (1998): *Auditoría Informática. Un Enfoque Práctico*. Ra-Ma, Madrid (España).
- Piattini, M., Polo, M., Ruiz, F. y Calero, C. (1999a): Utilización de los Estándares ISO/IEC en el Mantenimiento del Software. *V International Congress on Information Engineering (ICIE'99)*. Buenos Aires (Argentina). Cataldi et al (editores), pp. 1-8.
- Piattini, M., Polo, M., Ruiz, F. y Calero, C. (1999b): A Rigorous Approach for Software Maintenance. *Fourth World Conference on Integrated Design & Process Technology (IDPT'1999)*. Dallas (Estados Unidos). Society for Design and Process Science, CD-ROM.
- Piattini, M., Ruiz, F., Polo, M., Villalba, J., Bastanchury, T., Martínez, M.A. y Nistal, C. (2000): *Mantenimiento del Software: Modelos, Técnicas y Métodos para la Gestión del Cambio*. Ra-Ma (España).
- Pigoski, T.M. (1997): *Practical Software Maintenance. Best Practices for Managing your Investment*. John Wiley & Sons, Estados Unidos.
- Pinto, H.S. y Martins, J.P. (2001): Ontology Integration: How to perform the Process. *Workshop on Ontologies and Information Sharing* (dentro del IJCAI2001). Seattle (Estados Unidos). AAAI Press, pp. 71-80.
- Plexousakis, D. (1995): Simulation and Analysis of Business Processes Using GOLOG. *Proceedings of the Conference on Organizational Computing Systems (COOCS'95)*, pp. 311-323.
- PMI (2000): *PMBOK: A Guide to the Project Management Body of Knowledge*, 2000 edition. Project Management Institute Communications, Estados Unidos.
- Podnar, I., Mikac, B. y Caric, A. (2000): SDL Based Approach to Software Process Modeling. En Conradi, R. (editor); *7th European Workshop on Software Process Technology (EWSPT'2000)*. LNCS 1780, Springer-Verlag, pp. 190-202.

- Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999a): *MANTEMA versión 2.0: una Metodología para el Mantenimiento de Software*. Informe Técnico UCLM-DI-99-01, Universidad de Castilla-La Mancha, Departamento de Informática, Grupo Alarcos, 91 pg.
- Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999b): Roles in the Maintenance Process. *ACM Software Engineering Notes*; 24(4), pp. 84-86.
- Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999c): MANTEMA: A Complete Rigorous Methodology for Supporting Maintenance based on the ISO/IEC 12207 Standard. Third *Euromicro Conference on Software Maintenance and Reengineering* (CSMR'99). Amsterdam (Países Bajos). IEEE Computer Society, pp. 178-181.
- Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999d): Mantema: A Software Maintenance Methodology based on the ISO/IEC 12207 Standard. 4th *IEEE International Software Engineering Standards Symposium* (ISESS'99). Curitiba (Brazil). IEEE Computer Society, pp. 76-81.
- Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999e): Adaptación de la Norma ISO/IEC 12.207 al Proceso de Mantenimiento del Software. 6º *Congreso Internacional de Investigación en Ciencias Computacionales* (CIIC'99). Cancún (México), pp. 92-100.
- Polo, M. (2000): *MANTEMA: Una Metodología para el Mantenimiento del Software*. Universidad de Castilla-La Mancha, Departamento de Informática; tesis doctoral, 19-Julio-2000.
- Polo, M., Piattini, M. y Ruiz, F. (2000a): Planning the non-Planneable Maintenance. *11th European Software Control and Metrics Conference / 3rd Scope Conference on Software Product Quality* (ESCOM-SCOPE 2000). Munchen (Germany). Shaker Publishing, pp. 49-57.
- Polo, M., Piattini, M. y Ruiz, F. (2000b): Cuestionario para la Identificación de Riesgos en Proyectos de Mantenimiento. *V Jornadas de Ingeniería del Software y Bases de Datos* (JISBD'2000). Valladolid (España), pp. 47-52.
- Polo, M., Garbajosa, J., Piattini, M. y Ruiz, F. (2000c): Métodos y Técnicas para la Mejora del Proceso de Mantenimiento. *VII Congreso Internacional de Nuevas Tecnologías y Aplicaciones Informáticas*. La Habana (Cuba). MIC, CD-ROM.
- Polo, M., Mateos, M.D., Piattini, M. y Ruiz, F. (2001a): Distributing Human Resources among Software Development Projects. *12th European Software Control and Metrics Conference* (ESCOM '2001). Londres (Reino Unido), 2-4 abril.
- Polo, M., Piattini, M. y Ruiz, F. (2001b): Using Code Metrics to Predict Maintenance of Legacy Programs: a Case Study. *IEEE International Conference on Software Maintenance* (ICSM'2001). Florence (Italia), pp. 202-208.
- Polo, M., Piattini, M., Ruiz, F. y Jiménez, M. (2001c): Assessment of Maintenance Maturity in IT Departments of Public Entities: Two Case Studies. *Product Focused Software Process Improvement* (PROFES'2001). Springer-Verlag, LNCS 2188, ISBN 3-540-42571-3, pp. 86-97.
- Polo, M., Piattini, M. y Ruiz, F. (2001d): MANTOOL: a Tool for Supporting the Software Maintenance Process. *Journal of Software Maintenance and Evolution: Research and Practice*. 13(2), pp. 77-95, John Wiley & Sons.
- Polo, M., Piattini, M. y Ruiz, F. (2001e): Experience adapting ISO/IEC 12207 to the Maintenance Process. En "*Software Engineering – Software standardization*". DinTel, España, pp. 165-189.
- Polo, M., Piattini, M. y Ruiz, F. (2002a): Using a Qualitative Research Method for Building a Software Maintenance Methodology. *Software Practice & Experience*., 32(13), pp. 1239-1260. John Wiley & Sons.
- Polo, M., Piattini, M. y Ruiz, F. (2002b): Integrating Outsourcing in the Maintenance Process. *Information Technology and Management*, Vol 3(3), pp. 247-269. Kluwer Academic Publishers.
- Polo, M., Piattini, M. y Ruiz, F. [editores] (2003): *Advances in Software Maintenance Management: Technologies and Solutions*. Idea Group Publishing (Estados Unidos).
- Poole, C.J., Murphy, T., Huisman, J.W. y Higgins, A. (2001): Extreme Maintenance. *IEEE International Conference on Software Maintenance* (ICSM'01). 7-9 noviembre, Florencia (Italia), pp. 301-311.

- Pressman, R.S. (1998): *Ingeniería del Software. Un Enfoque Práctico* (4ª edición). McGraw-Hill Interamericana, España.
- Rajlich, V.T. y Bennett, K.H. (2000): A Staged Model for the Software Life Cycle. *IEEE Computer*, July 2000, pp. 66-71.
- Ramage, M. y Bennett, K. (1998): Maintaining Maintainability. *International Conference on Software Maintenance* (ICSM). 16-19 marzo, Bethesda, Maryland (Estados Unidos), pp. 275-283.
- Ramanujan, S., Scamell, R.W. y Shah, J.R. (2000): An Experimental Investigation of the Impact of Individual, Program, and Organizational Characteristics on Software Maintenance Effort. *Journal of Systems and Software*, 54(2), pp. 137-157.
- Ramaswamy, R. (2000): How to Staff Business-Critical Maintenance Projects. *IEEE Software*, 17(3), pp. 90-94.
- Randall, R. y Ett, W. (1995): "Using Process to Integrate Software Engineering Environments". Proceedings of the *Software Technology Conference*, Salt Lake City, USA.
- Rasmus, D.W. (1996): Mind Tools: Connecting to Groupware. *PC AI*, September/October, pp 32-36.
- Rational Software Corporation (1997): *UML 1.1 Extension for Objectory Process for Software Engineering*. Disponible en <http://www.rational.com/uml/resources/documentation/formats.jsp>.
- Reimer, W., Schgfer, W. y Schmal, T. (1997): Towards a Dedicated Object Oriented Software Process Modelling Language. En *Object-Oriented Technology - ECOOP '97 Workshop Reader*. LNCS 1357, Springer-Verlag, pp. 299-302.
- Ribó, J.M. y Franch, X. (2001): Building Expressive and Flexible Process Models using a UML-based Approach. Proceedings of the 8th. European Workshop in Software Process Technology (EWSPT-01). Witten (Germany). LNCS 2077, Springer-Verlag, pp. 152-172.
- Rock-Evans, R. (1993): *Repositories and Frameworks: a Detailed Product Evaluation*, Vol. 1, Ovum Ltd.
- Rombach, H.D. y Verlage, M. (1995): "Directions in Software Process Research". En Zelkowitz, M.V. (editor), *Advances in Computers*, Vol. 41, Academic Press, Boston, USA, pp. 1-61.
- Royce, W. (1993): Why Software Costs so Much?. *IEEE Software*, 10(3), pp. 90-91, May 1993.
- Ruiz, F., Piattini, M., y Polo, M. (1998): MANTEMA: Una Metodología para la Gestión Integral del Mantenimiento del Software. VI Encuentro Chileno de Computación (ECC'98). Antofagasta (Chile), pp. 184-193.
- Ruiz, F., Piattini, M., Polo, M. y Calero, C. (1999a): Auditoría del Mantenimiento del Software: Propuesta de Objetivos de Control. *II Congreso Nacional de Auditoría y Control de Sistemas de Información* (CASI'99). Valencia (España), octubre. pp. 128-143.
- Ruiz, F., Piattini, M., Polo, M. y Calero, C. (1999b): Propuesta de un Marco Formal para la Auditoría del Proceso de Mantenimiento del Software. *6º Congreso Internacional de Investigación en Ciencias Computacionales* (CIIC'99). Cancún (México), septiembre. pp. 192-202.
- Ruiz, F., Piattini, M., Polo, M. y Calero, C. (1999c): Maintenance Types in the MANTEMA Methodology. *International Conference on Enterprise Information Systems* (ICEIS'99). Setubal (Portugal). Filipe y Cordeiro (editores), pp. 192-202.
- Ruiz, F., Piattini, M., Polo, C. y Calero, C. (2000a): Audit of Software Maintenance Process. En "Auditing Information Systems". Idea Group Publishing (Estados Unidos), pp. 67-108.
- Ruiz, F., Piattini, M. y Polo, M. (2000b): Objetivos de Control para la Auditoría del Proceso de Mantenimiento del Software. *V Seminario Iberoamericano de Protección contra Virus Informáticos y Seguridad de las TI*. La Habana (Cuba), CD-ROM.
- Ruiz, F. y Piattini, M. (2001): Entorno Global para la Gestión del Proceso de Mantenimiento del Software. *VI Jornadas de Ingeniería del Software y Bases de Datos* (JISBD'2001). Almagro (Ciudad Real), 21-23 noviembre, pp. 157-170.

- Ruiz, F., Piattini, M., García, F. y Polo, M. (2001a): Metamodelos y Flujos de Trabajo para la Gestión del Proceso de Mantenimiento del Software. *4ª Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes de Software (IDEAS'2001)*. Heredia (Costa Rica), abril 2001, pp. 132-142.
- Ruiz, F., Piattini, M., y Polo, M. (2001b): Using Metamodels and Workflows in a Software Maintenance Environment. *VII Argentine Congress on Computer Science (CACIC'01)*. El Calafate (Argentina), octubre 2001.
- Ruiz, F., Piattini, M., y Polo, M. (2001c): An Conceptual Architecture Proposal for Software Maintenance. *13th International Symposium on System Integration (ISSI'01)*. Baden-Baden (Germany), pp. VIII:1-8.
- Ruiz, F., García, F., Márquez, L., Piattini, M. y Polo, M. (2001d): Tool based on MOF for Software Process Metamodelling. *Business Information Technology Management (BitWorld'2001)*. El Cairo (Egipto), junio. CD-ROM.
- Ruiz, F., Piattini, M. y Polo, M. (2002a): An Integrated Environment for Managing Software Maintenance Projects. En *"The Guide to IT Service Management, volume I"*, chapter 31, pp. 567-588. Addison-Wesley, ISBN 0201737922.
- Ruiz, F., García, F., Piattini, M. y Polo, M. (2002b): Environment for Managing Software Maintenance Projects. En *"Advances in Software Maintenance Management: Technologies and Solutions"*. Idea Group Publishing (Estados Unidos), capítulo X, pp. 255-290. ISBN 1-59140-047-3.
- Ruiz, F., Piattini, M., García, F. y Polo, M. (2002c): An XMI-based Repository for Software Process Meta-modeling. *Product Focused Software Process Improvement (PROFES'2002)*. Rovaniemi (Finlandia), 9-11 diciembre. Springer-Verlag, LNCS 2559, ISBN 3-540-00234-0. pp. 546-558.
- Ruiz, F., Polo, M. y Piattini, M. (2002d): Utilización de Investigación-Acción en la Definición de un Entorno para la Gestión del Proceso de Mantenimiento del Software. *I Workshop en Métodos de Investigación y Fundamentos Filosóficos en Ingeniería del Software y Sistemas de Información (MIFISIS'2002)*. El Escorial (España), noviembre, pp. 48-64.
- Ruiz, F., Vizcaíno, A., Piattini, M. y García, F. (2003): An Ontology for the Management of Software Maintenance Projects. *International Journal of Software Engineering and Knowledge Engineering* (pendiente de evaluación).
- Rumbaugh, J., Jacobson, I. y Booh, G. (1999): *The UML Reference Manual*. Addison-Wesley.
- Russell, S.J. y Norvig, P. (1995): *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, New Jersey (USA).
- Sadiq, W. y Orlowska, M.E. (1996): *Modeling and Verification of Workflows Graphs*. University of Queensland, informe técnico nº 386, noviembre 1996, Brisbane (Australia).
- Sadiq, W. y Orlowska, M.E. (1999): On Capturing Process Requirements of Workflow Based Business Information Systems. *3rd International Conference on Business Information Systems (BIS '99)*. Poznan (Polonia), 14-16 abril. Springer-Verlag, pp. 195-209.
- Scheer, A.W. (1999): *ARIS - Business Process Frameworks* (third edition). Springer Verlag.
- Schleicher, A. (1999): Formalizing UML-based Process Models using Graph Transformations. En Nagl, M., Schürr, A (editores); *International Workshop on Applications of Graph Transformations with Industrial Relevance (AGTIVE'99)*. Castle Rolduc (Holanda), pp. 341-357.
- Schleicher, A. y Westfechtel, B. (2001): Beyond Stereotyping: Metamodeling Approaches for the UML. *34th Hawaii International Conference on System Sciences (HICSS'2001)*. Maui, Hawaii (USA), enero 2001. IEEE Computer, pp. 3051.
- Schlenoff, C., Gruninger M., Tissot, F., Valois, J., Lubell, J. y Lee, J. (2000): *The Process Specification Language (PSL): Overview and Version 1.0 Specification*. NISTIR 6459, National Institute of Standards and Technology, Estados Unidos. Disponible en <http://www.mel.nist.gov/psl/pubs.html>.

- Schneidewind, N.F. (1997): Measuring and Evaluating Maintenance Process Using Reliability, Risk and Test Metrics. 19th *International Conference on Software Maintenance (ICSM)*. Los Alamitos, California (Estados Unidos), IEEE Computer Society Press, pp. 232-239.
- Schneidewind, N.F. (1998): How to Evaluate Legacy Systems Maintenance. *IEEE Software*, 15(4), pp. 34-42.
- Schneidewind, N.F. (1999a): Software Quality Maintenance Model. *International Conference on Software Maintenance (ICSM'99)*. Oxford (Reino Unido), IEEE Computer Society Press, pp. 277-286.
- Schneidewind, N.F. (1999b): Measuring and Evaluating Maintenance Process Using Reliability, Risk, and Test Metrics. *IEEE Transactions on Software Engineering*. 25(6), pp. 769-781.
- Schneidewind, N.F. (2000): Software Quality Control and Prediction Model for Maintenance. *Annals of Software Engineering*. Vol 9, pp. 79-101.
- Schneidewind, N.F. (2001): Investigation of the Risk to Software Reliability and Maintainability of Requirements Changes. *IEEE International Conference on Software Maintenance (ICSM'01)*. 7-9 noviembre, Florencia (Italia), pp. 127-136.
- Schneidewind, N.F. (2002): Requirements Risk and Maintainability. En Polo, M., Piattini, M. y Ruiz, F. (editores); *Advances in Software Maintenance Management: Technologies and Solutions*. Idea Group Publishing (Estados Unidos), capítulo VII.
- Schürr, A. (1996): Introduction to the Specification Language PROGRES. En Nagl, M. (editor); *Building Tightly Integrated Software Development Environments: The IPSEN Approach*. LNCS 1170, Springer-Verlag, pp. 248-279.
- Seaman, C.B. (1999): Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering*, 25(4), 557-572.
- SEI (2002): *Capability Maturity Model Integration (CMMI)*, version 1.1. March 2002. Disponible en <http://www.sei.cmu/cmmi/cmmi.html>.
- Sellink, A. y Verhoef, C. (1999): An Architecture for Automated Software Maintenance. *Seventh International Workshop on Program Comprehension (IWPC)*. 5 - 7 mayo, Pittsburgh, Pennsylvania (Estados Unidos), pp. 38-48.
- Serrano, M.A. (2001): *Herramienta CASE para la Estimación de Esquemas de Bases de Datos*. Universidad de Castilla-La Mancha, Dep. de Informática. Proyecto fin de carrera, septiembre 2000.
- Sharp, A. y McDermott, P. (2001): *Workflow Modeling: Tools for Process Improvement and Application Development*. Artech-House.
- Shepherd, M., Schofield, C. y Kitchenham, B. (1996): Effort Estimation Using Analogy. *International Conference on Software Maintenance*. Los Alamitos, California (Estados Unidos). IEEE Computer Society Press, pp. 170-178.
- Sherer, S.A. (1997): Using Risk Analysis to Manage Software Maintenance. *Journal of Software Maintenance: Research and Practice*, 9(6), pp. 345-364.
- Sierra, R. (1994): *Análisis Estadístico Multivariable. Teoría y Ejercicios*. Madrid: Paraninfo.
- Silverman, B.G., Bedewi, N. y Morales, A. (1995): Intelligent Agents in Software Reuse Repositories. *CIKM Workshop on Intelligent Information Agents*. Baltimore, (Maryland (USA)).
- Slaughter, S.A. y Banker, R.D. (1996): A Study of the Effects of Software Development Practices on Software Maintenance Effort. *International Conference on Software Maintenance (ICSM '96)*. 4-8 noviembre, Monterey, California (Estados Unidos), pp. 197-205.
- Sneed, H.M. (2001): Impact Analysis of Maintenance Tasks for a Distributed Object-oriented System. *IEEE International Conference on Software Maintenance (ICSM'01)*. 7-9 noviembre, Florencia (Italia), pp. 180-189.
- Springer-Verlag (2002): *Springer LINK Search*. Disponible en <http://link.springer.de/search.htm>.

- Spyns, P., Meersman, R. y Jarrar, M. (2002): Data Modelling versus Ontology Engineering. *SIGMOD Record*, 31(4), December-2002, pp. 12-17.
- Stark, G.E. (1996): Measurements for Managing Software Maintenance. *International Conference on Software Maintenance* (ICSM '96). 4-8 noviembre, Monterey, California (Estados Unidos), pp. 152-161.
- Stoecklin, S., Williams, D. y Stoecklin, P. (1998): Tailoring the Process Model for Maintenance and Reengineering. *2nd Euromicro Conference on Software Maintenance and Reengineering* (CSMR). Florencia (Italia), pp. 209-212.
- Sutton, S.M., Tarr, P.L. y Osterweil, L.J. (1995a): *An Analysis of Process Languages*. TR 95-78, Computer Science Department, University of Massachusetts, USA.
- Sutton, S.M., Heimbigner, D. y Osterweil, L.J. (1995b): APPL/A. *ACM Transactions on Software Engineering and Methodology* (TOSEM), 4 (3), pp. 221-286.
- Sutton, S.M. y Osterweil, L.J. (1997): The Design of a Next-generation Process Language. *ACM SIGSOFT Software Engineering Notes*, 22(6), pp. 142-158.
- SWEBOK (2001): *Guide to the Software Engineering Body of Knowledge*; Stone Man Trial version 1.00 (mayo 2001). Software Engineering Coordinating Committee (IEEE-CS y ACM). Disponible en <http://www.swebok.org>.
- Tan, W-G. y Gable, G.G. (1998): Attitudes of Maintenance Personnel towards Maintenance Work: a Comparative Analysis. *Journal of Software Maintenance*, 10(1), pp. 59-74.
- Tautz, C. y Von Wangenheim, C.G. (1998): *REFSENO: A Representation Formalism for Software Engineering Ontologies*. Fraunhofer IESE-Report No. 015.98/E, version 1.1, October 20.
- Thayer, R.H. (1997): Software Engineering Project Management. En Thayer, R.H. (editor); *Software Engineering Project Management* (2nd edition). IEEE Computer, pp. 72-104.
- Thomas, I. y Nejme, B.A. (1992): Definitions of Tool Integration for Environments. *IEEE Software*, 9(2), pp. 29-35.
- Tomer, A. y Schach, S. (2000): The Evolution Tree: A Maintenance-Oriented Software Development Model. *Conference on Software Maintenance and Reengineering*. Zurich (Suiza), IEEE Computer Society, pp. 209-214.
- USAF (1996): Software Engineering Environment Integration Process – Summary-level Definition. *Software Technology for Adaptable, Reliable Systems (STARS)*, TR PV03-A032/001/00, abril 1996.
- Visaggio, G. (1999): Assessing the Maintenance Process Through Replicated, Controlled Experiments. *Journal of Systems and Software*. Elsevier Science, 44(3), pp. 187-197.
- Vizcaíno, A., Ruiz, F., Favela, J. y Piattini, M. (2002a): A Multi-Agent Architecture for Knowledge Management in Software Maintenance. I *International Workshop on Practical Applications of Agents and Multiagent Systems* (IWPAAMS'2002). Salamanca (España), 23-25 octubre.
- W3C (1998a): *Extensible Markup Language (XML), Version 1.0*. W3C Consortium. Disponible en <http://www.w3.org/TR/1998/REC-xml-19980210>.
- W3C (1998b): *Document Object Model (DOM) Level 1 Specification, version 1.0*. W3C Consortium. Disponible en <http://www.w3.org/DOM/>.
- W3C (1999): Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation 22 February 1999. Disponible en <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>
- W3C (2000): *Extensible Markup Language (XML) 1.0* (second edition). W3C Recommendation, octubre 2000. Disponible en <http://www.w3.org/XML/>.
- W3C (2001): XML Schema Part 0: Primer. W3C Recommendation, 2 May 2001. Disponible en <http://www.w3.org/TR/xmlschema-0/>.

- W3C (2003): Web Services Description Language (WSDL) version 1.2; working draft 3 march 2003. W3C Consortium. Disponible en <http://www.w3.org/TR/2003/WD-wsdl12-20030303/>.
- Wadsworth, Y. (1998): What is participatory Action Research?. *Action Research International*, paper 2. Disponible en <http://www.scu.edu.au/schools/sawd/ari/ari-wadsworth.html> (accedido el 20-marzo-2002).
- Wallnau, K.C. y Feiler, P.H. (1991): *Tool Integration and Environment Architectures*. Technical Report CMU/SEI-91-TR-11, May 1991.
- Wang, X. y Chan, C.W. (2001): Ontology Modeling Using UML. *7th International Conference on Object Oriented Information Systems Conference (OOIS'2001)*. Calgary (Canada), pp.59-68.
- Wang, Y. y King, G. (2000): *Software Engineering Processes: Principles and Applications*. CRC Press.
- Wasserman, A. (1989): Tool Integration in Software Engineering Environments. En Long, F. (editor), *Software Engineering Environments: Proceedings of the International Workshop on Environments*. Chinon (Francia), LNCS 467, Springer-Verlag, pp. 137-149.
- Weber, R. (1999): *Information Systems Control and Audit*. Prentice-Hall (Estados Unidos).
- Westfechtel, B. [editor] (1999a): *Models and Tools for Managing Development Processes*. LNCS 1646, Springer-Verlag.
- Westfechtel, B. (1999b): Toward and Adaptable Environment for Modeling and Managing Development Processes: Overview. En Westfechtel, B. (editor); *Models and Tools for Managing Development Processes*. LNCS 1646, Springer-Verlag, pp. 287-303.
- WfMC (1995): TC00-1003 1.1: *The Workflow Reference Model*. Workflow Management Coalition, enero 1995. Disponible en <http://www.wfmc.org/standards/docs.htm>.
- WfMC (1999): TC-1016-P 1.1: *Interface 1: Process Definition Interchange Process Model*. Workflow Management Coalition, octubre 1999. Disponible en <http://www.wfmc.org/standards/docs.htm>.
- WfMC (2002): TC-1025 final draft 1.0: *Workflow Process Definition Interface - XML Process Definition Language (XPDL)*. Workflow Management Coalition, octubre 2002. Disponible en <http://www.wfmc.org/standards/docs.htm>.
- Wood-Harper, T. (1985): Research Methods in Information Systems: Using Action Research. En Mumford et al. (eds.); *Research Methods in Information Systems*. Amsterdam: North-Holland, pp. 169-191.
- Xie, T. (2001): A Linguistic Study of Process Modeling Languages. Department of Computer Science & Engineering, University of Washington. CSE 505 Concepts of Programming Languages. Disponible en <http://www.cs.washington.edu/homes/taoxie/research.htm>.
- Zahran, S. (1997): *Software Process Improvement: Practical Guides for Business Success*. Addison-Wesley.
- Zelkowitz, M.V. (1993): Use of an Environment Classification Model. *Fifteenth ACM/IEEE International Conference on Software Engineering*, Baltimore (USA), pp. 348-357.
- Zelkowitz, M.V. (1996): Software Engineering Environment Capabilities. *Journal of Systems and Software*. Elsevier Science, 35(1), pp. 3-14.