[BROOK75] Brooks, F., The Mythical Man-Month, Addison-Wesley, 1975.

[CMU/SEI-91] Paulk, Mark C.; Curtis, Bill; & Chrissis, Mary Beth; Capability Maturity Model for Software (CMU/SEI-91-TR-24, ADA 240603). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, 1991.

[CMU/SEI-92-TR-22] Software Quality Measurement: A Framework for Counting Problems and Defects, Software Engineering Institute, Carnegie Mellon University, 1992.

[CORON00] Coronado, S., Advanced Integrated Software Program Management Methodology based on Software Processes, PhD Dissertation, Universidad Politécnica de Madrid, Spain June 2000.

[CSEP95] Computation Science Education Project, Introduction to Monte Carlo Methods, Sponsored by U.S. Department of Energy ,1995.

[GILKS96] Grey, S., Practical Risk Assessment for Project Management, John Wiley & Sons, 1995.

[IEEE-STD-610] IEEE Standard Glossary of Software Engineering Terminology IEEE 610.12-1990.

[KAN95] Kan, S., Metrics and Models in Software Quality Engineering, Addison Wesley, 1995.

[SOBOL94] Sobol', I., A Primer for the Monte Carlo Method, CRC Press 1994.

Entorno Global para la Gestión del Proceso de Mantenimiento del Software [1]

Francisco Ruiz*, Mario Piattini*

*Escuela Superior de Informática, Universidad de Castilla-La Mancha
Ronda de Calatrava, s/n. 13071, Ciudad Real (España)
{fruiz|mpiattin}@inf-cr.uclm.es

Abstract. El mantenimiento del software es un campo de investigación donde todavía quedan muchos aspectos por estudiar (Bennett y Rajlich, 2000). De igual forma, el desarrollo de herramientas y entornos para dar soporte a las metodologías y para facilitar la realización de los procesos, es uno de los aspectos básicos de investigación, de cara al futuro, dentro del campo de la ingeniería del software (Harrison et al, 2000). En este trabajo se presenta MANTIS, un entorno global para gestionar el proceso de mantenimiento del software que incluye tres tipos de herramientas: conceptuales, metodológicas y técnicas (software). La integración de todas estas herramientas en un único entorno es una ayuda para abordar la complejidad inherente al proceso de mantenimiento desde una perspectiva de proceso de negocio, más amplia que la meramente tecnológica. De todos los diferentes componentes de MANTIS, en este documento se hace hincapié en aquellos que más directamente tienen relación con el objetivo de ayudar a gestionar la complejidad: un marco conceptual que permite trabajar con modelos y metamodelos a diferentes niveles de abstracción; una extensión del metamodelo general del PMS para incluir también aspectos dinámicos de realización de los proyectos; y unas herramientas que permiten definir, almacenar y gestionar todos los conceptos empleados.

1 Introducción

Aunque tradicionalmente la comunidad de ingeniería del software ha considerado el proceso de mantenimiento mucho menos importante que el proceso de desarrollo, en los últimos años esta situación ha empezado a cambiar. Ejemplo de este cambio de tendencia es la reciente propuesta de Rajlich y Bennet (2000) de ciclo de vida del software. Para estos autores, desde una perspectiva de negocio, un producto software pasa por cinco etapas diferentes: *desarrollo inicial*, *evolución*, *servicio*, *retirada*, y *cierre*. Cada una de dichas etapas difiere de las demás en algunos aspectos esenciales, y por tanto tiene sus propias actividades, herramientas y consecuencias de negocio.

[1] Este trabajo ha sido parcialmente financiado con cargo a los proyectos MANTIS (CICYT-FEDER 1FD97-1608TIC) y MPM (TA15-1999, programa ATYCA del MINER y Atos ODS).

JISBD 2001
I.S.B.N.: 84-699-6275-2

Puesto que el proceso de mantenimiento del software (PMS) ocurre durante los cuatro últimas etapas citadas, es necesario disponer de algunos métodos y herramientas específicos, distintos de los disponibles para el proceso de desarrollo (Piattini et al, 2000). En este línea se encuentran aportaciones como MANTEMA (Polo et al, 1999), una metodología basada en el modelo de procesos software de ISO 12207 (ISO/IEC, 1995), o las técnicas de mejora del PMS propuestas por Niessink (2000).

Por otro lado, 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 Methodologie*". En el proyecto MANTIS (Ruiz et al, 2001) se ha aplicado esta misma idea para construir un entorno global que permita abordar la gestión de proyectos de mantenimiento de software con una perspectiva de negocio, incluyendo, entre otros, los aspectos siguientes:

- Las personas con ciertas habilidades desempeñan ciertos roles en el proyecto trabajando juntas en diversos equipos (grupos de 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.

El Entorno (con E mayúscula) MANTIS extiende e integra los conceptos de *Metodología* (en el sentido habitual, es decir, una serie de métodos o técnicas relacionados), y de *Entorno de Ingeniería del Software* (EIS), entendiendo este último como una colección de herramientas software utilizadas para soportar actividades de ingeniería del software (ISO/IEC, 2000). Sus principales características son:

a) Orientado al uso 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 con estos objetivos (PCTE, ECMA/NIST, STARS, ...) que incluyen contribuciones y modelos que pueden utilizarse de forma genérica en el PMS, ninguna de ellas tiene en cuenta las características especiales del mantenimiento.

b) Dirigido a procesos: La orientación a procesos es un vehículo de integración muy importante (Randall y Ett, 1995). En MANTIS se contemplan las formas de integración: de los procesos de la organización con el "Entorno", de las herramientas y artefactos con los procesos. Esta última es conocida como "*Process Sensitive Software Engineering Environment*" (Derniame et al, 1999). Además, en MANTIS se asegura la consistencia con los procesos actividades descritos en los estándares ISO 12207, de procesos del ciclo de vida del software , y 14764, del PMS (ISO/IEC, 1998).

c) Especializado en el mantenimiento: Utiliza MANTEMA como metodología básica, y se han desarrollado adaptaciones propias de diversos métodos y técnicas.

d) Escalable: La adaptación a las necesidades de proyectos de mantenimiento de software de cualquier tamaño se basa, fundamentalmente, en el uso de un marco conceptual que permite una gran genericidad. También se contempla la utilización de un "*tayloring process*" según es definido en la norma ISO 12207.

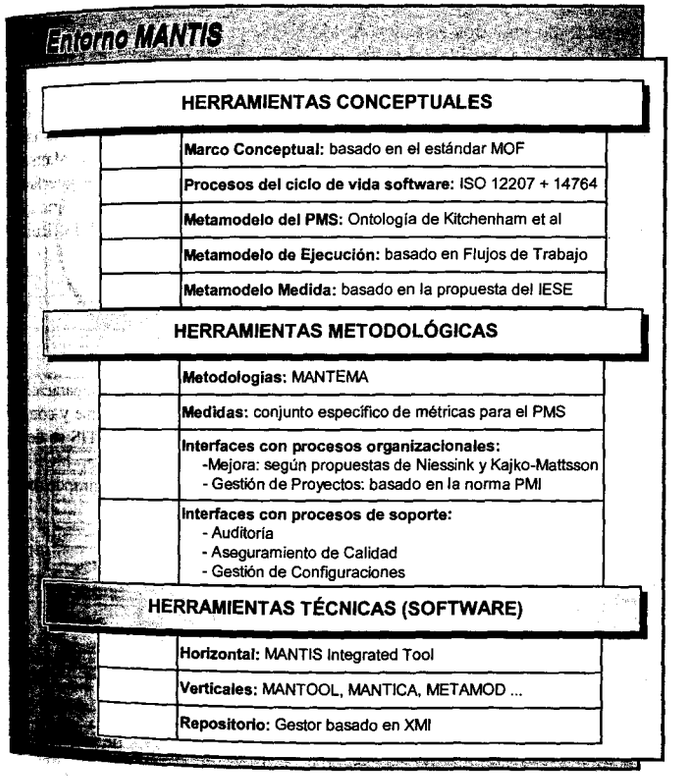

Fig. 1. Componentes del Entorno MANTIS.

En la figura 1 se muestra un resumen de los componentes del Entorno MANTIS. Todos los componentes se consideran herramientas; clasificadas en tres grandes categorías: conceptuales, metodológicas y técnicas-software (Ruiz et al, 2001). El uso

combinado de todas ellas ayuda a gestionar mejor el PMS. Por ello, el principal objetivo de MANTIS, es disponer de un entorno que permita integrar conceptual, metodológica e instrumentalmente estas herramientas.

A continuación se presentan los tres aspectos que influyen más directamente en que MANTIS sea útil para gestionar la dificultad de los proyectos de mantenimiento del software. En el apartado 2 se presenta el marco conceptual general. El metamodelo extendido para incluir la estructura interna de las actividades y los aspectos de ejecución de proyectos reales se comenta en el apartado 3. Las herramientas para gestión de metamodelos y del repositorio se incluyen en el apartado 4. Por último, en el apartado 5 se presentan algunas conclusiones y las líneas de trabajo futuras.

Respecto de los otros aspectos de MANTIS, merece resaltar que la integración del modelo de ciclo de vida del software definido en las normas ISO 12207 y 14764 se realiza dentro de la metodología MANTEMA. Igualmente, las métricas específicas para el PMS se incluyen como uno de los componentes de MANTEMA (Polo y Piattini, 1999). Los interfaces con los procesos organizacionales utilizan las propuestas de mejora del PMS formuladas por Niessink (2000) y Kajko-Mattson et al (2001), y el modelo PMBOK de gestión de proyectos (PMI, 2000). Los interfaces con los procesos de soporte están en desarrollo, salvo el del proceso de auditoría que está basado en la definición de 14 objetivos de control propios para el PMS (Ruiz et al, 2000).

2 Marco Conceptual

Un principio importante de la ingeniería del software moderna es la separación de un sistema en capas de encapsulación, que pueden especificarse, diseñarse y construirse de manera independiente en gran parte. Con esta filosofía, en MANTIS se definen 4 niveles conceptuales (ver tabla 1) que están basados en el estándar MOF (Meta Object Facility) para metamodelización con orientación a objetos, propuesto por el Object Management Group, OMG (2000).

Tabla 1. Niveles Conceptuales en MOF y MANTIS.

Nivel	MOF	MANTIS
M3	Modelo MOF (Meta-metamodelo)	Modelo MOF
M2	Meta-modelo	Metamodelo general del PMS
M1	Modelo	MANTEMA y otras técnicas (modelo concreto del PMS)
M0	Datos	Instancias de proyectos concretos de mantenimiento del software

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. El

concreto utilizado en el nivel M1 está basado en la metodología MANTEMA y en un conjunto de técnicas adaptadas a las particularidades del mantenimiento: estimación de esfuerzo, estimación de riesgos, etc (Polo et al, 1999). El nivel M2 se corresponde con el metamodelo general del PMS, que es generalizado mediante un modelo-MOF en el nivel M3. Un modelo-MOF está formado básicamente por dos tipos de objetos: MOF-class y MOF-association. Por tanto, todos los conceptos representados en el nivel M2 se consideran ejemplares de MOF-class o de MOF-association en el nivel M3. Por ejemplo, "Actividad de Mantenimiento", "Recurso" o "Artefacto" serán ejemplares de MOF-class y "Actividad usa Recurso" o "Artefacto es_entrada_de Actividad" son ejemplares de MOF-association. Este nivel permite llegar a la mayor abstracción ya que, aunque se trabaje con diferentes modelos y metamodelos de proceso software, todos ellos se pueden representar utilizando un modelo-MOF.

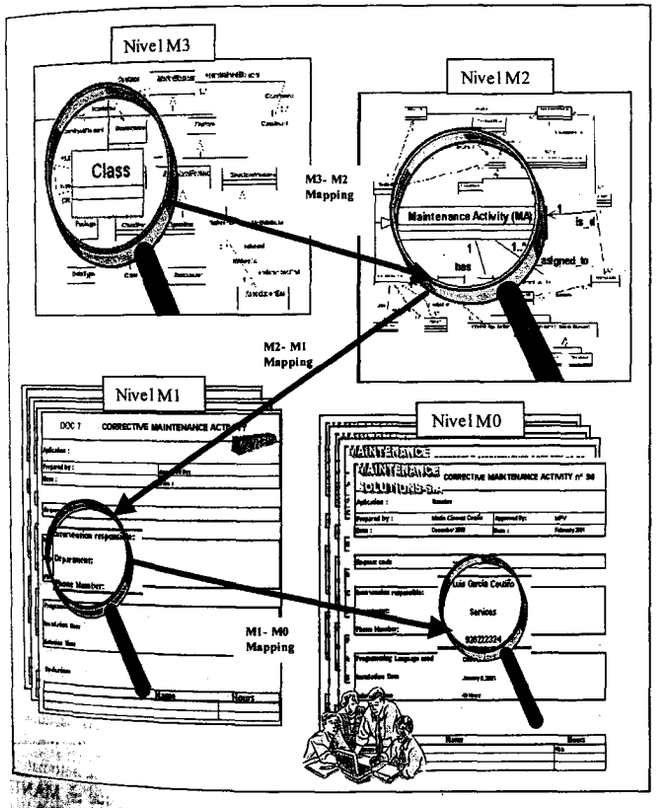

Fig. 2. Ejemplo de niveles conceptuales en MANTIS.

En la figura 2 se muestra un ejemplo con las correspondencias entre los 4 niveles: la "Intervención correctiva nº 36 en el proyecto PATON" (nivel M0) es un ejemplar de la clase "Actividad de Mantenimiento Correctivo" (nivel M1), que a su vez es un ejemplar de la clase más genérica "Actividad de Mantenimiento" (nivel M2); que también es un ejemplar de MOF-class.

El metamodelo genérico del PMS utilizado en el nivel M2 está basado en la propuesta de ontología informal para el mantenimiento del software formulada por Kitchenham et al (1999). Este metamodelo está construido por la unión de cuatro metamodelos parciales centrados en las actividades, los productos, las personas y el proceso (Ruiz et al, 2001b).

3 Extensiones del metamodelo

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 (Almeida et al, 1998). Para satisfacer este requisito en el Entorno MANTIS no basta con permitir diversos niveles de abstracción (utilizando el marco conceptual comentado); 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 entre 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 (Gruber, 1995).

Por las razones anteriores, un objetivo secundario de MANTIS es elaborar una ontología del PMS, común a todos los componentes del Entorno (figura 1). Para ello la ontología informal propuesta por Kitchenham et al (1999) está siendo ampliada y formalizada. Un cierto nivel de formalización es también necesario para poder representar la ontología mediante objetos de los niveles conceptuales ya comentados y también para poder construir las herramientas que almacenen y gestionen los modelos y metamodelos.

A continuación presentamos, a título de ejemplo, la "***ontología de los flujos de trabajo***", una de las extensiones del metamodelo genérico del PMS (nivel M2) realizada para incluir los aspectos dinámicos de ejecución (reificación) de los proyectos de mantenimiento y la descomposición interna de las actividades en actividades o tareas más simples.

Recientemente, algunos autores (Ocampo y Botella, 1998) han sugerido la posibilidad de utilizar Flujos de Trabajo (FT) para abordar los procesos software sacando partido de la similitud existente entre ambas tecnologías. Este planteamiento ha sido confirmado por algunos desarrollos concretos de herramientas basadas en FT para dar soporte al proceso software en general (Penades et al, 1999). 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 esta perspectiva más amplia, en línea con los objetivos del Entorno MANTIS.

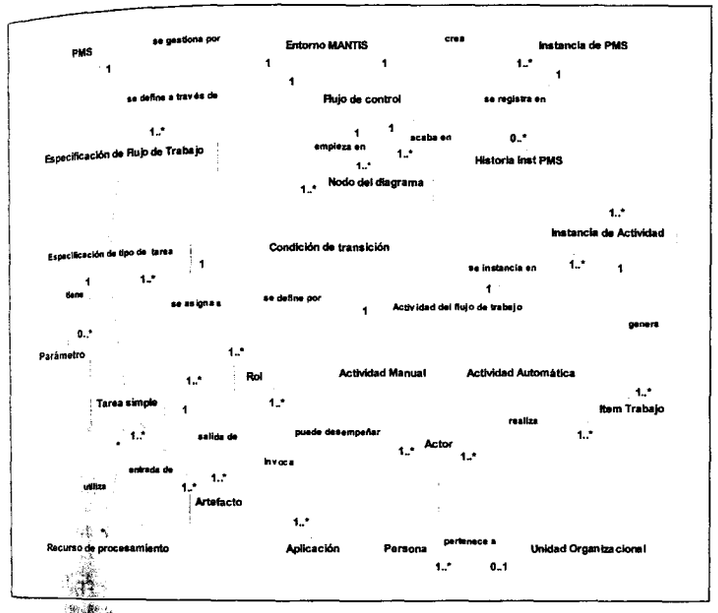

Fig. 3. Resumen de la "*Ontología de los flujos de trabajo*" de MANTIS (nivel M2).

En la figura 3 se muestra un resumen de la ontología de los FT de MANTIS, que ha sido desarrollada a partir del modelo de referencia propuesto por la *Workflow Management Coalition* (WfMC, 1995) y de la propuesta de Liu et al (1999). En esta ontología se han incluido dos aspectos principales: la especificación de la estructura de actividades y subactividades y sus relaciones (parte izquierda de la figura); y la información necesaria para administrar y controlar la ejecución del PMS (parte derecha de la figura).

3.1 Especificación de las Actividades

La ontología de la figura 3 representa los siguientes objetos e interrelaciones:

- El "PMS" propiamente dicho, que es visto como un proceso de negocio gestionado usando el "Entorno MANTIS".

- La clase "Especificación de tipo de tarea" contiene las propiedades que se pueden abstraer de una actividad o tarea [2] y que son independientes de los flujos de trabajo, es decir, incluye las propiedades generales de un conjunto de tareas del mismo tipo. Se distingue entre cada "Especificación de tipo de tarea" y el nodo que la representa en el FT ("Actividad del FT").
- Una "Especificación de tipo de tarea" puede ser atómica o anidada. Las primeras no tienen estructura interna y se representan mediante la clase "Tarea simple". Las segundas tienen una estructura interna que es representada utilizando una "Especificación de FT". La ejecución de una tarea anidada supondrá la ejecución del FT subyacente en la misma.
- Se utiliza recursividad para representar la jerarquía de tareas. Un proyecto de mantenimiento se modela mediante una única "Especificación de tipo de tarea" principal. Esta tarea principal tiene asociado un FT que incluye diversas "Actividades de FT" (que aparecen como nodos del diagrama de flujo), cada una de las cuales se define por una "Especificación de tipo de tarea". A su vez, cada una de estas especificaciones tiene asociada una "Especificación de FT" que incluye otras "Actividades de FT", y así sucesivamente. El número de niveles de la recursividad viene dado por los niveles utilizados en la estructura de descomposición de trabajos del proyecto. La recursividad acaba cuando una "Especificación de tipo de tarea" corresponde a una "Tarea simple".
- Cada "Especificación de FT" es representada mediante las clases "Nodo del diagrama", "Flujo de control", "Condición de transición" y "Actividad del FT". El modelo de FT utilizado está basado en la propuesta de Sadiq y Orlowska (1999).
- Una "Especificación de FT" se representa utilizando diagramas de flujo, es decir, un conjunto de "Nodos del diagrama" interconectados mediante "Flujos de control" (flechas). Los "Nodos del diagrama" pueden ser "Actividades del FT" o "Condiciones de Transición". 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 concurrencia) o más de un flujo de control acabando en ellas (final de concurrencia o sincronización).
- Para contemplar que pueden llevarse a cabo actividades automatizadas, la clase "Actividad de un FT" tiene las especializaciones "Actividad Manual" y "Actividad Automática". Las diferencias entre ambas son las características en tiempo de ejecución. Una "Actividad de FT" puede pertenecer simultáneamente a ambas subclases. Esta posibilidad permite la existencia

[2] Aunque las palabras "Tarea" y "Actividad" se pueden utilizar como sinónimos, se ha optado por utilizar "Actividad" para los nodos de los FT (siguiendo la nomenclatura de la WfMC) y "Tarea" para la representación de la descomposición de trabajos del proyecto (siguiendo la nomenclatura de las normas ISO).

actividades mixtas, que son realizadas en parte de forma manual y en parte de forma automática.

- Los "Actores" que intervienen en el proceso pueden ser "Personas" o "Unidades organizacionales" a las que pertenecen las personas. Cada "Actor" puede desempeñar unos determinados "Roles". Para contemplar la posibilidad de que ciertas tareas se realicen de manera automática o semiautomática invocando aplicaciones externas, se incluye la especialización "Aplicación" de "Actor". Estas "Aplicaciones" pueden ser invocadas desde "Tareas simples".
- La clase "Parámetro" permite que cada "Especificación de tipo de tarea" pueda tener asociados varios parámetros. El propósito de estos parámetros es gestionar la información que es recibida o producida por el correspondiente tipo de tarea.
- Cada "Especificación de tipo de tarea" puede ser asignada a varios "Roles" diferentes. Cada "Rol" puede ser desempeñado por diversos "Actores".
- El seguimiento y control de un proyecto se realiza a nivel de sus tareas simples. Por esta razón, los "Recursos de procesamiento" que utiliza una tarea y los "Artefactos" de entrada o de salida se asocian con "Tareas simples".

3.2 Reificación del Proceso

Los aspectos dinámicos, relacionados con la ejecución, se representan mediante los siguientes objetos e interrelaciones:

- Al llevar a cabo un proyecto concreto de PMS, el Entorno MANTIS crea una "Instancia de SMP", que además se registra en un "Histórico de Instancias de SMP".
- Durante la realización del proyecto, se crean una o varias "Instancias de Actividad" para cada "Actividad del FT". La opción de poder crear varias instancias de ejecución de una misma "Actividad" está incluida para contemplar iteraciones (bucles).
- Cada "Instancia de Actividad" genera uno o varios "Items de trabajo" (porción más pequeña de trabajo que es realizada, gestionada y controlada). Cada "Item de trabajo" es realizado por uno o varios "Actores".

4 Herramientas

Una de las características fundamentales del Entorno MANTIS es su orientación a las herramientas. Por esta razón, además de incluir actividades automatizadas en la ontología de los flujos de trabajo, se facilita el uso integrado de todas las herramientas software al seguir los principios arquitecturales de la propuesta PSEE (Derniame et al, 1999).

- Utilizar un motor de proceso (process engine) para controlar el flujo de información entre los actores o realizadores, de acuerdo con el modelo de proceso.
- Almacenar en un repositorio el metamodelo del proceso junto con la definición del producto (el software) y la información de reificación del proceso.
- 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.

En MANTIS existen dos tipos de herramientas software: una herramienta horizontal, llamada MANTIS-Tool, cuyo objetivo es automatizar la gestión integral de proyectos de mantenimiento de software de forma similar a un PSEE; y varias herramientas verticales, cada una de las cuales sirve para automatizar alguno de los tipos de actividad que forman parte del PMS.

4.1 MANTIS-Tool

El objetivo específico de MANTIS-Tool es ayudar a la gestión global de los proyectos de mantenimiento permitiendo la mayor automatización posible y la integración de la información y las herramientas utilizadas. Para ello es importante que permita representar todos los aspectos comentados en la introducción y que se pueda comunicar con otras herramientas, externas (SGFT, CASE, etc.) o verticales incluidas en el Entorno MANTIS. MANTIS-Metamod es el componente que se encarga de lo primero: permite definir y editar modelos y metamodelos de procesos software e instancias de ejecución de proyectos concretos del mundo real que reifican los modelos de proceso anteriores (Ruiz et al, 2001c). En suma, permite trabajar con los niveles del marco conceptual de MANTIS.

Un proyecto de mantenimiento de software genera datos de tres clases diferentes de *Producto* (código fuente, de gestión de versiones y de configuración, documentación, ejecutables, conjuntos de pruebas, resultados de pruebas, etc.), de *Proceso* (definición explícita del metamodelo del PMS, información del estado de la ejecución del proceso, para análisis y evolución, históricos, de gestión del proyecto, etc.); y *Organizacionales* (propietarios de los entregables del proyecto, roles y responsabilidades, información de equipos de trabajo, de gestión de recursos, etc.). MANTIS-Repository es el componente de MANTIS-Tool encargado de almacenar y recuperar todos estos datos y metadatos (García et al, 2001). Para garantizar la máxima flexibilidad e intercambiabilidad con otras herramientas y entornos, este gestor de repositorio utiliza el formato "*XML Metadata Interchange*" (XMI) propuesto por OMG (2000b).

4.2 Herramientas verticales

El Entorno MANTIS incluye diversas herramientas específicas para automatizar algunos de los tipos de actividades que forman el PMS: seguimiento de peticiones de

modificación, estimación del esfuerzo de mantenimiento, control de riesgos, etc. Entre estas herramientas, la más significativa es MANTOOL, que sirve para gestionar las peticiones de modificación (PM) de acuerdo con la metodología MANTEMA (Polo et al, 2001).

MANTOOL permite realizar el seguimiento del conjunto de PM asociadas a una cartera de proyectos. Cada PM se asigna a un tipo mantenimiento (entre los cinco diferentes definidos en MANTEMA). La figura 4 muestra la pantalla correspondiente para el caso de mantenimiento correctivo urgente: aparece un grafo con las cinco tareas a realizar y seis pestañas para poder incluir los datos de cada tarea y generales de la PM. Todos estos datos se almacenan en formato XMI para permitir su uso compartido por todas las herramientas del Entorno MANTIS.

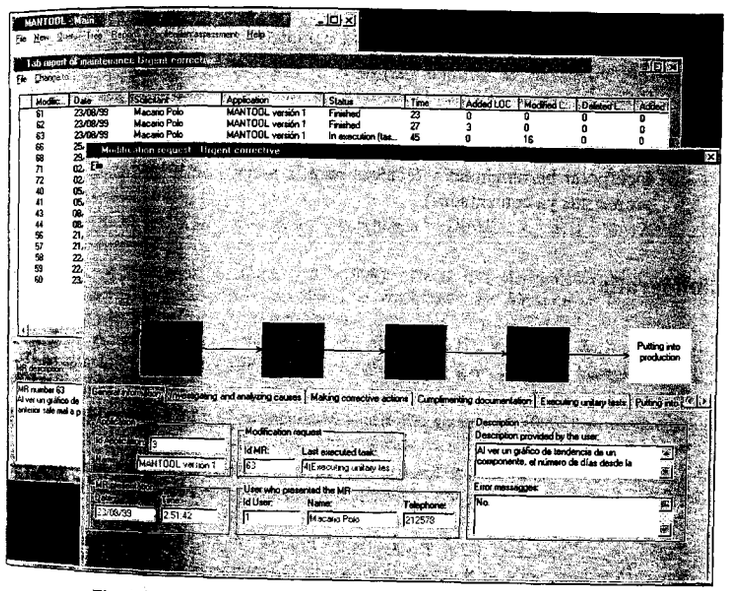

Fig. 4. Seguimiento de peticiones de modificación con MANTOOL.

5 Conclusiones y trabajos futuros

El mantenimiento es la etapa más costosa del ciclo de vida de un producto software. Por ello, es importante disponer de "Entornos de Ingeniería del Software" que ayuden a gestionar proyectos de mantenimiento de software de la mejor manera posible. Estos EIS deben cumplir dos objetivos básicos: poder compartir toda la información generada y utilizada, y poder utilizar de forma integrada las diversas herramientas

disponibles. En este trabajo se ha presentado el Entorno MANTIS, que intenta satisfacer los objetivos comentados, abordando el PMS con una perspectiva amplia de proceso de negocio. MANTIS pretende ser una ayuda para gestionar la complejidad inherente al PMS, concediendo especial importancia a:

1. La integración del conocimiento: utilizando ontologías para especificar de manera precisa y formal el dominio del problema, y empleando un marco conceptual (basado en la norma MOF) con 4 niveles de abstracción diferentes.
2. Una arquitectura orientada a las herramientas (propuesta PSEE) junto con tecnologías que facilitan la integración y compartición de los datos y metadatos.

En la actualidad se está trabajando para completar el Entorno MANTIS. Entre otros, los trabajos en desarrollo más significativos son:

- Elaborar una ontología de la medida que permita incorporar todos los aspectos de métricas, medición y valores. También se pretende construir una herramienta vertical para dar soporte automático al proceso de medición.
- Incorporar herramientas y técnicas para la mejora del PMS (basadas en las propuestas ya comentadas).

Bibliografía

1. Bennett, K. & Rajlich, V. (2000): "Software Maintenance and Evolution: A Roadmap". *International Conference on Software Engineering* (ICSE) - Future of SE Track, pp. 73-87.
2. Cockburn, A. (2000): "Selecting a Project's Methodology". *IEEE Software*, July/August, pp. 64-71.
3. Derniame, J.C., Kaba, B.A., & Wastell, D. (1999): "The Software Process: Modelling and Technology". In *Software Process: Principles, Methodology and Technology*. LNCS 1500, Springer-Verlag.
4. Falbo, R.A., Menezes, C.S., Rocha, A.R. (1998): "Using Ontologies to Improve Knowledge Integration in Software Engineering Environments". Proceedings of the 4th *International Conference on Information Systems Analysis and Synthesis*, SCI'98/ISAS'98, Orlando (USA).
5. García, F., Ruiz, F., Piattini, M., Márquez, L. y Polo, M. (2001): "Propuesta de Repositorio basada en XMI para metamodelado de procesos software". XXVII *Conferencia Latinoamericana de Informática*. Mérida (Venezuela).
6. 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.
7. Harrison, W., Ossher, H. & Tarr, P. (2000): "Software Engineering Tools and Environments: A Roadmap". *International Conference on Software Engineering* (ICSE) - Future of SE Track, pp. 261-277.
8. ISO/IEC 12207 (1995): *Information Technology – Software Life Cycle Processes*.

9. ISO/IEC FDIS 14764 (1998): *Software Engineering - Software Maintenance* (draft), Dec-1998.
10. ISO/IEC JTC1/SC7/WG4 15940 working draft 5 (2000): *Information Technology – Software Engineering Environment Services*, Juny-2000.
11. Kitchenham, B.A., Travassos, G.H., Mayrhauser, A., Niessink, F., Schneidewind, N.F., Singer, J., Takada, S., Vehvilainen, R. & Yang, H. (1999): "Towards an Ontology of Software Maintenance". *Journal of Software Maintenance: Research and Practice*. 11, pp. 365-389.
12. Liu, C.; Lin, X.; Zhou, X.; 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.
13. Kajko-Mattson, M., Forssander, S., Olsson, U. (2000): "Corrective Maintenance Madurity Model: Maintainer's Education and Training". *International Conference on Software Engineering* (ICSE), Toronto (Canada).
14. Niessink, F. (2000): "*Perspectives on Improving Software Maintenance*". PhD Thesis, Vrije Universiteit, Netherland. In http://www.opencontent.org/openpub/.
15. Ocampo, C. y Botella, P. (1998): "*Some Reflections on Applying Workflow Technology to Software Processes*". TR-LSI-98-5-R, UPC, Barcelona, 1998.
16. OMG (2000): "*Meta Object Facility (MOF) Specification*", v. 1.3 RTF. In http://www.omg.org.
17. OMG (2000b): "*XML Metadata Interchange*" (XMI), v. 1.1, Nov-2000. In http://www.omg.org.
18. Penadés, M.C.; Canós, J.; Carsí, J.A. (1999): "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).
19. Piattini, M.; Ruiz, F.; Polo, M., Bastanchury, T., Fernández, I., Martínez, M.A. (2000): "*Mantenimiento del Software: Conceptos, Métodos, Herramientas y Outsourcing*". Ed Ra-Ma, Madrid.
20. PMI (2000): "*PMBOK: A Guide to the Project Management Body of Knowledge*" 2000 edition. Project Management Institute Communications, USA.
21. Polo, M. & Piattini, M. (1999): "Elaboración de una Metodología para el Mantenimiento del Software". IV *Jornadas de Ingeniería del Software y Bases de Datos* (JISBD'99), Cáceres (España), pp. 219-230.
22. Polo, M., Piattini, M., Ruiz, F. and Calero, C. (1999): "MANTEMA: A Complete Rigourous Methodology for Supporting Maintenance based on the ISO/IEC 12207 Standard". Third *Euromicro Conference on Software Maintenance and Reengineering* (CSMR'99). IEEE Computer Society Press, Amsterdam (Netherland), pp. 178-181.
23. Polo, M., Piattini, M. y Ruiz, F. (2001): "MANTOOL: a tool for supporting the software maintenance process". *Journal of Software Maintenance and Evolution: Research and Practice*. Vol 13(2), pp. 77-95.
24. Rajlich, V. & Bennett, K.H. (2000): "A Staged Model for the Software Life Cycle". *IEEE Computer*, July 2000, pp. 66-71.
25. Randall, R. & Ett, W. (1995): "Using Process to Integrate Software Engineering Environments". Proceedings of the *Software Technology Conference*, Salt Lake City, USA.
26. Ruiz, F., Piattini, M., Polo, M. y Calero, C. (2000): "Audit of Software Maintenance". In *Auditing Information Systems*. Idea Group Publishing, USA, pp. 213-223.

27. Ruiz, F., Piattini, M. & Polo, M. (2001): "An Integrated Environment for Managing Software Maintenance Projects". In *World Class IT Service Management Guide* (2nd edition), Addison-Wesley (publicación prevista en noviembre-2001).
28. Ruiz, F., Piattini, M., Garcia, F. y Polo, M. (2001b): "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*. San José (Costa Rica), pp. 132-142.
29. Ruiz, F., Garcia, F., Marquez, L., Piattini, M., y Polo, M. (2001c): "Tool based on MOF for software process metamodelling". *Business Information Technology Management Conference*. El Cairo (Egipto).
30. WfMC TC00-1003 1.1 (1995): *The Workflow Reference Model*. Workflow Management Coalition. Jan-1995.

Esquema de Caracterización para la Selección de Técnicas de Pruebas

Óscar Dieste Tubío
Escuela Politécnica Superior.
Universidad Alfonso X
Campus de Montegancedo s/n.
28691 Villanueva de la Cañada, Madrid, España
odiestub@uax.es

Sira Vegas Hernández
Facultad de Informática. UPM
Campus de Montegancedo
28660 Boadilla del Monte, Madrid, España
svegas@fi.upm.es

Resumen. Uno de los problemas más importantes que existen en el área de pruebas del software, es la obtención de un conjunto de casos de prueba adecuados para probar un sistema software. Dicho conjunto, debe ser tal que su efectividad sea máxima, con un número mínimo de casos. Actualmente, hay numerosas técnicas de pruebas disponibles para la obtención de casos de prueba. Sin embargo, algunas son mal utilizadas, mientras otras no se usan nunca, y solamente unas pocas son las que se usan continuamente. Cuando los desarrolladores tienen que decidir las técnicas de pruebas que deben utilizar en un proyecto, tienen poca (si alguna) información acerca de las técnicas disponibles, su utilidad, y en general, cómo de adecuadas son al proyecto actual. En este artículo se presentan los resultados de desarrollar un artefacto (al que se ha denominado esquema de caracterización) para la selección de técnicas de pruebas. Cuando se instancia para diversas técnicas, el esquema proporciona a los desarrolladores información suficiente para que elijan las técnicas más apropiadas para el proyecto en el que están trabajando. De este modo, se consigue que las decisiones que toman estén basadas en conocimiento fundamentado sobre las técnicas en lugar de percepciones, suposiciones y asunciones.

1 Introducción

Las pruebas de software consisten en ejecutar un sistema software para un conjunto de entradas. Los principales objetivos de las pruebas de software son: (1) encontrar defectos o (2) evaluar un determinado atributo de calidad del software (corrección, fiabilidad, usabilidad, etc.). Un problema en las pruebas de software es la elección de un conjunto adecuado de entradas al sistema (casos de prueba) que muestren el comportamiento deseado del sistema en detalle, ya la vez sea lo suficientemente reducido para no malgastar el tiempo.