JIISIC '08

www.jiisic08.espol.edu.ec

VII Jornadas Iberoamericanas

de Ingeniería de Software e Ingeniería del Conocimiento

Memoria Técnica

30 de Enero al 1 de Febrero

Guayaquil - Ecuador

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL ESPOL

INGENIERÍA DE SOFTWARE

VLIR Vlaamse Interuniversitair Raad UOS University Development Co-operation

VII Jornadas Iberoamericanas

de Ingeniería de Software e Ingeniería del Conocimiento

Comité Organizador

Mónica Villavicencio Cabezas, Escuela Superior Politécnica del Litoral del Ecuador (chair).
María Verónica Macías Mendoza, Escuela Superior Politécnica del Litoral del Ecuador.
Carlos Monsalve Arteaga, Escuela Superior Politécnica del Litoral del Ecuador.

Grupo Editor de la Revista

Mónica Villavicencio - Directora
Lohana Lema Moreta - Asistente

Colaboradores de edición

Stephanie Flores
Guillermo Pizarro

Diseño de portada

Luis Bajaña

Colaboradores

Emilio Rigazio
David Jurado
Karina Chong
Fátima Cedeño

JIISIC'08

VII Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento

Guayaquil – Ecuador
Del 30 de Enero al 1 de Febrero del 2008

Editado y Compilado por:
Escuela Superior Politécnica del Litoral
Facultad de Ingeniería Eléctrica y Computación
Área de Ingeniería en Software VLIR –ESPOL Componente 8

VII Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento

Compilado por:
Escuela Superior Politécnica del Litoral
Facultad de Ingeniería Eléctrica y Computación
Área de Ingeniería en Software VLIR –ESPOL Componente 8

Editado por:
Mónica Villavicencio, Carlos Monsalve, Verónica Macías, Guillermo Pizarro, Lohana Lema y Stephanie Flores.

Primera Edición: enero 2008

Jornadas de Ingeniería de Software (ISSN 1390-292X) será publicada cada vez que se organice unas Jornadas por el Área de Ingeniería de Software del Componente 8 del Proyecto VLIR-ESPOL.

Comité Permanente

Silvia Teresita Acuña, Universidad Autónoma de Madrid, España
Manoel Mendonça, Universidad Salvador, Brasil
Oscar Dieste, Universidad Complutense de Madrid, España
José Antonio Pow-Sang, Universidad Católica de Perú, Perú

Comité Organizador

Mónica Villavicencio Cabezas, Escuela Superior Politécnica del Litoral del Ecuador (chair)
María Verónica Macías Mendoza, Escuela Superior Politécnica del Litoral del Ecuador
Carlos Monsalve Arteaga, Escuela Superior Politécnica del Litoral del Ecuador

Comité de Programa

Aguilar Raúl, Universidad Autónoma de Yucatán - México
Álvarez Luis Alberto, Universidad Austral - Chile
Álvarez Marco, Utah State University, EEUU - USA
Antúnes Pedro, Universidad de Lisboa - Portugal
Aveledo Marianela, Universidad Simón Bolívar - Venezuela
Carrizo Dante, Universidad Complutense de Madrid - España
Chiluiza Katherine, Escuela Superior Politécnica del Litoral - Ecuador
De Antonio Angélica, Universidad Politécnica de Madrid – España
De Castro Valeria, Universidad Rey Juan Carlos - España
Duran Amador, Universidad de Sevilla - España
Echagüe Juan Vicente, Universidad de la República - Uruguay
Eterovic Yadran, Pontificia Universidad Católica de Chile - Chile
Fernández Mariano, Universidad CEU San Pablo - España
Ferre Xavier, Universidad Politécnica de Madrid - España
García Ramon, Instituto Tecnológico de Buenos Aires – Argentina

Comité del Programa (continuación)

García Francisco José, Universidad de Salamanca - España
Gómez Marta, Universidad CEU San Pablo - España
Grimán Anna, Universidad Simón Bolívar - Venezuela
Guerrero Luis, Universidad de Chile - Chile
Imbert Ricardo, Universidad Politécnica de Madrid - España
Jino Mario, Universidad Estadual de Campinas - Brasil
Kong Maynard, Pontificia Universidad Católica del Perú – Perú
Macías María Verónica, Escuela Superior Politécnica del Litoral - Ecuador
Macías José Antonio, Universidad Autónoma de Madrid - España
Muñoz Jaime, Universidad Autónoma de Aguascalientes - México
Pérez Melvin, CAM Informática - República Dominicana
Pons Claudia, Universidad Nacional de la Plata – Argentina
Pow-Sang José Antonio, Pontificia Universidad Católica del Perú - Perú

Prólogo

La presente publicación contiene el compendio de 49 artículos presentados y aprobados por el comité del programa de las VII Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento, las mismas que se desarrollaron en Guayaquil, Ecuador del 30 de Enero al 1 de febrero del 2008.

JIISIC es un evento que ha logrado posicionarse y mantenerse a nivel internacional, acogiendo a activos investigadores y profesionales interesados en presentar los resultados de sus trabajos de investigación, garantizando así su difusión y promoviendo el conocimiento.

En esta oportunidad 84 artículos de calidad científica fueron recibidos para su evaluación, de los cuales 49 fueron aceptados luego de 3 revisiones. Los autores de los artículos representaron a universidades y empresas de Argentina, Brasil, Colombia, Cuba, Chile, Ecuador, España, México, Perú, Uruguay y Venezuela. Adicionalmente, el evento contó con la presentación de 4 tutoriales y 4 conferencias magistrales.

Es justo aprovechar esta oportunidad para agradecer a quienes colaboraron incansablemente en el desarrollo de JIISIC 2008; entre ellos debo de mencionar ante todo a Lohana Lema por su entrega ilimitada, luego a Guillermo Pizarro, Luis Bajaña, Verónica Macías, Stephanie Flores y al grupo de voluntarios de la FIEC y de rama estudiantil de la IEEE. Un agradecimiento muy especial a José Antonio Pow-Sang por su invalorable y desinteresada ayuda. Finalmente, agradezco a todos los autores por habernos escogido y participar de este evento.

Mónica Villavicencio
Presidente del Comité Organizador

INDICE

Sección I: Base de Datos y Minería de Datos

TariyKDD: una herramienta de minería de datos débilmente acoplada con un SGBD
Ricardo Timarán Pereira, Andrés O. Calderón Romero, Iván Ramírez Freyre, Fernando Guevara, Juan Carlos Alvarado 3

El Modelo de Negocios Decisional como origen de Especificación de Requisitos en Proyectos de Data Mining: Una Aproximación Metodológica mediante el Framework i*
José Gallardo, Óscar Marbán, Claudio Meneses y Aldo Quelopana 13

Un Ambiente de Explotacion de Información basado en la Integración de Agrupamiento, Inducción y Ponderación Bayesiana de Reglas
G. Schulz, E. Fernández, H. Merlino, D. Rodríguez, P. Britos, R. García-Martínez 21

Mineria de Datos Aplicada a la Detección de Patrones Delictivos en Argentina
F. Valenga, E. Fernández, H. Merlino, D. Rodríguez, C. Procopio, P. Britos y R. García-Martínez 31

Paralelización de Consultas del tipo time-interval en Base de Espacio-Temporales
Claudio Gutiérrez-Soto, Gilberto Gutiérrez, Pedro Rodríguez, Pedro Campos 41

Sección II: Calidad, Procesamiento y Administración de Software

Sección II-a:

Incidencia de la planificación y las inspecciones en el desarrollo de proyectos de software: Caso de Estudio realizado en Ecuador
Raúl González Carrión, Mónica Villavicencio Cabezas 51

Factores de Éxito o Fracaso para la Mejora de Procesos Software: Caso Real en un Grupo de MiPyMEs
César Pardo, Julio Ariel Hurtado Alegria, Francisco J. Pino 59

Una Experiencia de Implantación de COMPETISOFT en una Pequeña Empresa Desarrolladora de Software
Jackson Mogrovejo, Abraham Dávila 67

Experiencia de Implementación de Mejora de Procesos en dos PYMEs Desarrolladoras de Software, que poseen certificación ISO 9001:2000
Gonzalo Sánchez, Dianne Vergara, Abraham Dávila 73

Sección II-b:

PROCODI: Lenguaje de Extensibilidad para UML
Daniel Alberto Giulianelli, Rocío Andrea Rodríguez, Pablo Martín Vera 81

Modelado del proceso de software con enfoque de negocio. Aplicacion de los estandares BPMN y UML
Mabel del V. Sosa, Silvia T. Acuña, Juan de Lara 89

Modelado de Mejora de Procesos de Software en Pequeñas Organizaciones
Ing. Pedro E. Colla, Dr. Jorge Marcelo Montagna 97

Sección II-c:

Métricas de Madurez en Conceptualización de Sistemas Expertos
M. Pollo-Cattaneo, Fernández, H. Merlino, D. Rodríguez, P. Britos, R. García-Martínez 107

Un Análisis Crítico Comparativo de Modelos y Estándares Relacionados con la Adquisición de Software
Gloria Piedad Gasca Hurtado, Gonzalo Cuevas Agustín 117

Hacia la definición de un modelo para la Gestión de Proyectos en el Desarrollo Global del Software: reflexiones sobre la situación actual
Miguel Ángel Blanco, Ismael Caballero, Mario Piattini 125

Sección II-d:

PROMETEU - a tool to support documents generation and traceability in the test process
Jorge Luiz da Cruz, Mario Jino, Adalberto Nobiato Crespo, Miguel Argollo 133

Mapeo de los Procesos de RUP respecto a MoProSoft
Katia Cánepa y Abraham Dávila 139

Analysis of an Artifact Oriented Test Process Model and of Testing Aspects of ISO/IEC 15504
Paulo M. S. Bueno, Adalberto N. Crespo, Clenio F. Salviano1, Mario Jino 147

Sección III: Diseño y Desarrollo de Software

Agilidad y disciplina en el Proceso de Desarrollo de Software para PyMES y Cooperativas en Latinoamérica: CASO VENEZUELA
George Di Paula, Dakar Parada, Maria Pérez, Luís Mendoza 157

Evolución del proceso de desarrollo de videojuegos en la Iniciativa Académica EDUMÓVIL
Gabriel Gerónimo-Castillo, Carlos Alberto Fernández-y-Fernández, Ricardo Ruiz-Rodríguez 163

Extensión a WSIL para la búsqueda y descubrimiento de servicios Web con calidad
Jesús Cruz-Ahuactzi, Giner Alor-Hernández, Ruben Posada-Gomez, Juan Miguel Gomez 171

Un modelo de arquitectura para el aprendizaje y compartición de conocimiento entre sistemas inteligentes autónomos distribuidos
Ierache, J., Naiouf, M., García Martínez, R., De Giusti, A. 179

Intérprete y Entorno de Desarrollo para el Aprendizaje de Lenguajes de Programación Estructurada
Layla Hirsh Martínez 189

Especificación Formal de Elementos MoProSoft a partir del Modelo de Referencia de Flujos de Trabajo
Leonel Valenzuela Ruiz, Brenda Leticia Flores Rios 197

Sección IV: Educación en Ingeniería de Software e Ingeniería del Conocimiento

Residência em Fábrica de Software: Um Caso Real e uma Proposta Genérica para a Normatização de Novos Programas
José Augusto Fabri, André Luiz Presende Trindade, Marcelo S. de Paula Pessôa 207

Un Modelo de Evaluación Adaptativa del Nivel de Conocimientos en Sistemas Tutoriales Inteligentes
Marcela Jiménez, Jovani A. Jiménez, Demetrio A. Ovalle 215

Sistema Tutor Inteligente con Tecnología de Agentes: La Elección del Método de Enseñanza

Zulma Cataldi, Patricia Calvo, Fernando J. Lage 223

TONGO: Un laboratorio para apoyar la experimentación sobre arquitecturas orientada a servicios
Jorge Villalobos, Fabián Contreras 231

Interacción con la Computadora: Modelo de Capacidades de Estudiantes de Nivel Medio de Jujuy
Viviana E. Quincoces, Héctor P. Liberatori, Ma. del Pilar Gálvez Díaz, Nilda M. Pérez Otero, Sandra A. Méndez, Adelina García, Cecilia Lasserre, Claudio M. Pérez Ibarra, Beatriz Fiorito 239

Sección V: Herramientas y técnicas de Software

Meta^{2}Relational: Herramienta para la Gestión de Modelos de Procesos Software
Tomás Martínez-Ruiz, Félix García, Mario Piattini 251

(Re)Composición de modelos ER con Idioms
Juan Marcelo Flores Soliz. Pablo Azero Alcocer 259

UN-LENCEP: A Controlled Language for Pre-conceptual Schema Specification
Carlos Mario Zapata Jaramillo, Alexander Gelbukh, Fernando Arango Isaza 269

Una Arquitectura Flexible para la Administración de Reputación en Sistemas Multiagentes
Víctor Daniel Podberezski, Jorge Salvador Ierache, Ramón García Martínez 277

INDIGO: una Propuesta de Planificación en Inteligencia Artificial para la Composición de Servicios Web Semánticos
Jaime Alberto Guzmán Luna, Demetrio Arturo Ovalle Carranza 287

Arquitecturas para Gestión de Conversaciones B2B Basadas en Conocimiento
José Luis López-Cuadrado, Juan Miguel Gómez, Angel García Crespo, Belén Ruiz Mezcua, Israel González-Carrasco, Giner Alor-Hernández 297

Una propuesta para el tratamiento de brotes epidémicos y de otros fenómenos espacio-temporales
Francisco Javier Moreno Arboleda 305

Aplicação de um Checklist de Pré-Teste
Odair Jacinto da Silva, Adalberto Nobiato Crespo, Mario Jino 313

Monitoring and Control of an Event-based Middleware
Oscar González, Nicolás López, and Rubby Casallas 319

Sección VI: Ingeniería del Conocimiento

Planificación de tareas en un Sistema de Computación Grid, para aplicaciones paralelas y/o secuenciales
Francisco Guevara, Andrés Marín, Elisa Heymann 335

Sistema de E-Learning basado en Agentes de Software, Sistemas Tutoriales Inteligentes, Ambientes Colaborativos de Aprendizaje
Andrés F. Hoyos P., Jovani A. Jiménez B. y Demetrio A. Ovalle C. 343

Adaptación, Recuperación y Almacenamiento de Contenidos Educativos Digitales para un Sistema Tutorial Inteligente
Catalina Salazar Ortiz, Jovani A. Jiménez B., Demetrio A. Ovalle C. 353

Sección VII: Mantenimiento y Reúso de Software

A Portlet-Based Service-Oriented Architecture for a second generation portal
Giner Alor-Hernández, Ruben Posada-Gomez, Juan Miguel Gomez, Ana Ma. Chávez-Trejo 363

Robots y Juguetes Autónomos una Oportunidad en el Contexto de las Nuevas Tecnologías en Educación
J. Ierache, M. Bruno, M Dittler, N. Mazza 371

Un enfoque ADM para la Reingeniería de Bases de Datos Relacionales hacia Servicios Web
García-Rodríguez de Guzmán, I., Polo, M., Piattini, M., Pérez, R. 1 Alarcos Research Group. 381

Anexo

Un enfoque pragmático para la mejora de procesos software en las PyMEs
Hanna Oktaba, Mario Piattini, Francisco J. Pino, Félix García, Tomás Martínez, Claudia Alquicira, Francisco Ruiz 395

Experiences with the use of MERODE in the development of a Web Based Application
Karina Chong, Verónica Macías, Monique Snoeck 421

Esquema de Clasificación de Defectos para la mejora del Proceso Software en una Empresa de Telecomunicaciones de Ecuador.
Fernando Uyaguary U. 427

Un enfoque ADM para la Reingeniería de Bases de Datos Relacionales hacia Servicios Web

[1]García-Rodríguez de Guzmán, I., [1]Polo, M., [1]Piattini, M., [2]Pérez, R.
[1]Alarcos Research Group.
UCLM-INDRA Research and Development Institute.
University of Castilla-La Mancha
Paseo de la Universidad, nº4 13071 – Ciudad Real (España)
{Ignacio.GRodriguez, Macario.Polo, Mario.Piattini}@uclm.es
[2]INDRA Sistemas
Ronda de Toledo s/n 13003 – Ciudad Real (España)
Ricardo.Perez3@alu.uclm.es

Resumen

Actualmente, la tecnología software está evolucionando a una gran velocidad debido a avances tecnológicos tales como SOA (Service Oriented Architecture) y los Servicios Web. El paradigma SOA esta basado en (1) el desarrollo del software como un conjunto de servicios y (2) la gestión de los sistemas heredados y otros artefactos mediante un mecanismo estandarizado, exponiendo sus funcionalidades como servicios. Por otro lado, paradigmas como MDA (Model-Driven Architecture) permiten a los ingenieros del software tratar con el software abstrayéndose de sus complejidades tecnológicas inherentes. ADM (Architecture-Driven Modernization) surge como una evolución del concepto de MDA, donde el proceso de reingeniería se desarrolla empleando modelos para representar todos los elementos involucrados en el proceso. De este modo, y siguiendo la secuencia sistema heredado→representación abstracta →nuevo sistema, en este artículo se presenta un completo proceso de reingeniería a nivel de modelo para la integración, como un conjunto de servicios, de bases de datos relacionales en entornos SOA.

Palabras Claves: Base de datos relacional, MDA, QVT, Architecture-Driven Modernization, SOA, reingeniería.

Abstract

Nowadays, software technology is evolving very quickly according to new trends in technology, such as SOA (Service Oriented Architecture) and Web Services. The SOA paradigm claims (1) to develop software as a set of services and (2) to handle legacy applications and artefacts using standardized technology, exposing their functionalities as services. On the other hand, paradigms like MDA (Model-Driven Architecture) allow software engineers to deal with software in a comfortable way, abstracting it from technological complexity. As a derivation of the MDA concept, Architecture Driven Modernization (ADM) focuses the reengineering process using models as first order artefacts. In this way, and following the chain legacy system-to-abstract representation-to-new system, we propose a complete process to reengineer relational databases at a model level to integrate them into SOA contexts as a set of services.

1. Introducción

Actualmente, los sistemas software están evolucionando rápidamente para así mantener la competitividad de las empresas. En ocasiones, los ciclos de desarrollo cortos y rápidos producen artefactos software pobres y mal estructurados [1]. Generalmente, la migración del software se lleva a cabo para permitir la evolución del mismo, aunque se presentan algunos problemas: (1) la heterogeneidad del software dentro de los sistemas de información [2], (2) sistemas desarrollados con tecnologías heredadas de diferentes épocas [3], y (3) sistemas que no disponen de la infraestructura necesaria para ser migrados a la Web [4]. Este problema actualmente se está agravando, ya que con el crecimiento del mercado global de servicios, la migración hacia la Web se está convirtiendo en algo esencial [5].

Hoy en día SOA (*Service Oriented Architecture*) puede considerarse como una de las tendencias tecnológicas más influyentes de los últimos tiempos. Frente a los paradigmas arquitectónicos previos, SOA ofrece la funcionalidad del sistema en forma de servicios y los hace accesibles a través de la Web. En SOA no sólo los nuevos desarrollos se plantean como servicios, sino también adaptar los sistemas heredados para que puedan ser expuestos de este modo, bien mediante técnicas de wrapping [6] u otros mecanismos.

Por otro lado, MDA [7] (*Model-Driven Architecture*) se está convirtiendo en un estándar maduro, haciendo surgir otros paradigmas estrechamente relacionados, como ocurre con el ADM [8, 9] (*Architecture-Driven Modernization*). ADM se puede ver como un reflejo del concepto de reingeniería, pero a nivel de modelo. Por esto, en cualquier proceso basado en ADM se asume que los sistemas heredados y los sistemas sometidos reingeniería son PSMs (*Platform Specific Models*), y las representaciones abstractas de los sistemas son PIMs (*Platform Independent Models*). En nuestro caso, tanto el PSM como el PIM pueden considerarse de la siguiente manera:

- PSM: un modelo que representa un sistema mediante una terminología ligada a una tecnología (ej. empleando un metamodelo de base de datos).
- PIM: un modelo que representa un sistema mediante una terminología no ligada a una tecnología específica, de manera que a posteriori (en una etapa posterior y empleando transformaciones adecuadas) el sistema pueda ser representado en base a una tecnología concreta (ej. Servicios Web o aplicaciones Java generadas a partir de diagramas UML).

Sin embargo, dependiendo del contexto y la personas involucradas, los conceptos de PSM y PIM pueden ser entendidos o situados en distintos niveles de abstracción. Incluso es posible distinguir varios niveles de abstracción dentro del PIM o el PSM. Por esto, quizá sea recomendable redefinir qué es un PSM o un PIM en cada proyecto basado en MDA que se emprenda.

Siguiendo las bases de ADM, el proceso propuesto en este trabajo se centra en el análisis de sistemas heredados para el descubrimiento de servicios y su publicación como Servicios Web. Esta propuesta está centrada en bases de datos relacionales (concretamente aquellas que implementan el estándar SQL-92 [10]), por su valor y su nivel de implantación en la industria [11, 12]. En este contexto, el PSM serían los metamodelos de base de datos SQL-92 y el metamodelo de WSDL (*Web Service Description Language*), y el PIM serían los metamodelos de UML2 [13] y OCL2 [14], que describen la base de datos y los servicios.

Este artículo esta organizado en las siguientes secciones: en la Sección 2 se incluyen algunos trabajos relacionados con los principales tópicos de esta propuesta; la Sección 3 describe el proceso propuesto; la Sección 4 muestra un caso de estudio llevado a cabo para validar la aproximación y, en la Sección 5 se detallan las conclusiones y algunas líneas de trabajo futuro.

2. Estado del arte

Es posible encontrar algunos trabajos relacionados con la reingeniería de bases de datos, concretamente la ingeniería inversa puede llevarse a cabo con distintos motivos [15-17]: redocumentación, migración de modelos, reestructuración, mantenimiento o mejora, requisitos tentativos, validación, integración, conversión de datos heredados, y verificación del estado del arte.

Es posible encontrar distintos algoritmos para la extracción y conceptualización de los metadatos de las bases de datos. Por ejemplo, en [18-20] los autores estudian algoritmos para la extracción de información sobre la estructura de bases de datos relacionales. J.L. Hainaut [21] ofrece también un estudio muy profundo sobre la ingeniería inversa de bases de datos.

DB-MAIN [22, 23] es una metodología genérica soportada por una herramienta con el mismo nombre Esta metododología esta compuesta de los siguientes pasos (grosso modo): (1) extracción de la estructura de la base de datos, y (2) conceptualización de las estructuras de datos.

El entorno MIDAS [15] trata el problema de la migración de bases de datos en red hacia bases de datos relacionales. Este marco permite también el reemplazo de subrutinas de acceso a la base de datos por código SQL.

Otra forma de migración puede ser el *wrapping*. Cuando un software debe utilizar una base de datos con un modelo de datos diferente, se puede crear una capa lógica entre el software y la base de datos, esto se conoce como *wrapper*. Los *wrappers* permiten transformar consultas para un modelo de datos en concreto en otro distinto [24]. Además, los *wrappers* pueden utilizarse para adaptar bases de datos relacionales a entornos distribuidos [25].

Tradicionalmente, la investigación en la reingeniería de bases de datos ha estado encaminada a tareas como la migración, la reestructuración, etc. Sin embargo, no existe mucho trabajo dentro de la aplicación de la reingeniería para la obtención de servicios a partir de bases de datos relacionales. En [26] se propone una completa herramienta para la generación automática de aplicaciones Web y de escritorio que permiten un acceso a la información de la base de datos.

Por otro lado, MDA ha probado ser una herramienta potente en muchas áreas de la ingeniería del software. ADM, como evolución del estándar MDA, mejora las capacidades de la arquitectura dirigida por modelos para tratar con procesos de reingeniería que empleen los modelos como artefacto básico en el mismo. En [9], OMG establece una "*hoja de ruta"* para el desarrollo de un marco completo compuesto de diversos estándares, metamodelos y métricas, y estandarizar así el concepto de ADM. Además, OMG identifica cuáles son los escenarios de aplicación más comunes para el ADM [27]. También se ha definido un estándar para la representación del conocimiento en cualquier proceso ADM [28].

En los ámbitos académico e industrial es también posible también encontrar trabajos que siguen la filosofía ADM [29, 30].

3. Proceso de reingeniería

El proceso propuesto consta (a alto nivel) de cinco pasos:

1. *Ingeniería inversa de la base de datos*: se extraen los metadatos del catálogo de la base de datos y se construye con ellos un modelo conforme a un metamodelo. Este modelo se considera como el PSM.
2. *Primera extracción de servicios*: Basándose en el esquema de la base de datos, se realiza una extracción de servicios mediante búsqueda en modelos o *Model-Driven Pattern Matching* (MDPEM) [31, 32].
3. *Generación del PIM*: a partir del PSM, se genera el PIM correspondiente, representado mediante UML. El PIM se obtiene mediante una transformación QVT [33]. Las operaciones CRUD y de acceso son incluidas automáticamente.
4. *Descubrimiento de servicios*: Se identifican *objetos abstractos* en el PIM. Para estos objetos se describen operaciones asociadas. Estas operaciones están descritas mediante modelos creados con el metamodelo de OCL.
5. *Generación de las interfaces WSDL*: A partir del PIM (y empleando una transformación QVT), se genera un conjunto de especificaciones WSDL expresadas como modelos. Estas especificaciones representan los Servicios Web que expondrán los servicios de la base de datos.

En las siguientes subsecciones se detallarán estos pasos, así como los artefactos involucrados en cada uno de ellos.

3.1. Ingeniería inversa de la base de datos

Este primer paso es fácil de automatizar ya que el estándar SQL-92 [10] propone un esquema común para el acceso a los metadatos: el INFORMATION_SCHEMA. El PSM es una instanciación de un metamodelo para PSM (un metamodelo para bases de datos relacionales esbozado en [34], y adaptado de [35]) que representa el esquema de la base de datos. Este primer paso se ha automatizado mediante el entorno *RelationalWeb* [26], un entorno para la reingeniería que permite generar distintos tipos de aplicaciones a partir de una base de datos relacional.

3.2. Primera extracción de servicios

Dado que el esquema de la base de datos describe completamente la estructura de la base de datos, el PSM describe la base de datos en sí con gran precisión. De este modo, es posible aprovechar la existencia de todos los elementos del metamodelo para realizar una primera extracción de servicios.

Para esta etapa, es necesario realizar una búsqueda de patrones a nivel de modelo (del PSM concretamente). Esta búsqueda se denomina *Model Driven Pattern Matching* o MDPEM (ver [31, 32] para más detalles). La idea del MDPEM es la de poder especificar un patrón en el que se involucren distintos elementos de la base de datos (como tablas, claves ajenas, columnas, etc.) para detectar determinadas estructuras con un comportamiento ya asociado.

Así, esta primera extracción de servicios se puede dividir en dos etapas: (1) selección de patrones, y (2) ejecución del MDPEM.

3.2.1. Selección de patrones. Basándose en la estructura de la base de datos es posible definir patrones que permitan buscar coincidencias en un esquema de base de datos dado.

Los patrones son también modelos, por lo que es necesario utilizar un metamodelo de patrón (ver Figura 1) compatible con la representación del PSM (en el contexto de QVT, el lenguaje para realizar la búsqueda de patrones) [33].

Siguiendo esta idea, en la Tabla 1 se proponen tres patrones básicos (a, b y c), más uno (c) descubierto tras la realización de un caso de estudio con una base de datos industrial:

a. *FK*: Patrón *foreign key*, dos tablas se relacionan mediante una clave ajena.
b. *DFK*: Patrón *double foreign key*, se involucran tres tablas, donde la tabla ***M*** tiene claves ajenas hacia las tablas ***A*** y ***B***.
c. *NFK*: Patrón *n-ary foreign key*, este patrón se considera una extensión del patrón DFK. Se considera que existen *n* claves ajenas desde ***M*** hacia *n* tablas (T_1, T_2, T_3 y T_n).
d. *NFK2ONE*: Este patrón representa una situación en la que *n* claves ajenas, que comienzan en *n* tablas (T_1, T_2, T_3 y T_n en la Tabla 1 (c)) y referencian a la misma tabla ***M***.

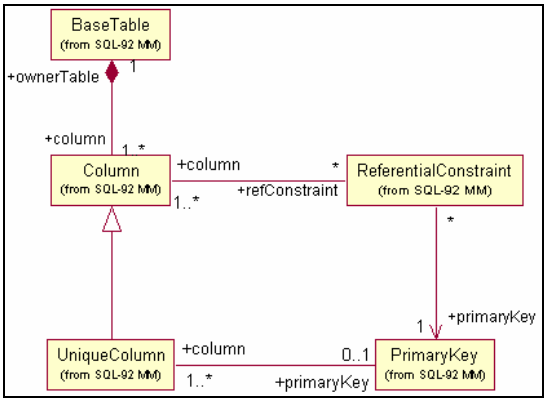
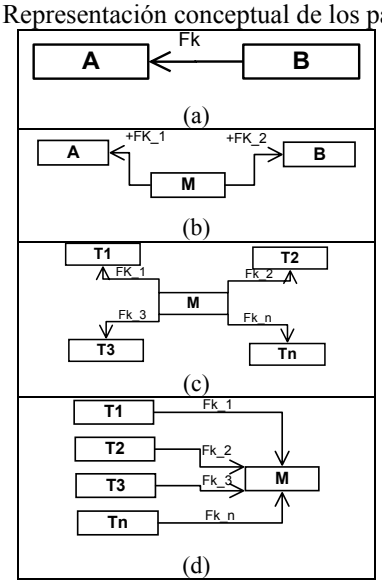


Figura 1. Metamodelo para la definición de patrones que permitan la búsqueda de ocurrencias en el PSM

Tabla 1. Representación conceptual de los patrones



Para cada patrón es posible definir operaciones genéricas expresadas en término a las tablas que componen el patrón. Tal y como se puede ver en la Figura 2, estas operaciones se especifican utilizando OCL. La Figura 2 (arriba) representa una expresión OCL, mientas que la Figura 2 (abajo) representa la misma expresión como un modelo conforme al metamodelo de OCL.

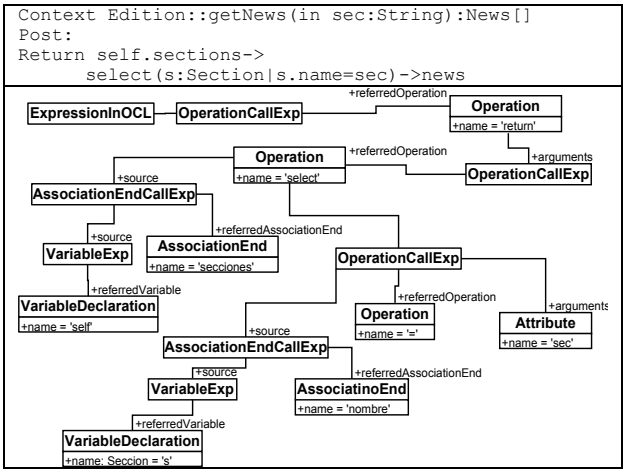


Figura 2. (arriba) Expresión OCL definiendo una operación y (abajo) su representación como modelo

3.2.2. Ejecución del MDPEM. Una vez que los patrones a aplicar han sido seleccionados por el ingeniero se ejecuta el MDPEM. La Tabla 2 describe todos los elementos involucrados en el MDPEM, mostrados según su nivel de abstracción en la Figura 3. Ver [31, 32] para más detalles.

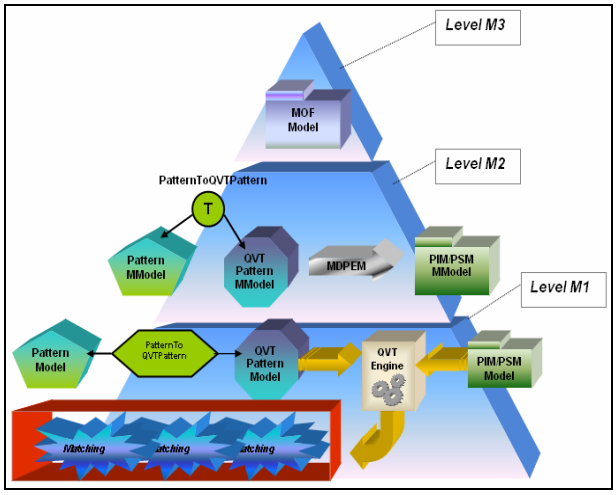


Figura 3. Marco basado en QVT para el MDPEM

El MDPEM se puede llevar a cabo en cuatro pasos: (1) definición de los modelos patrón y PSM/PIM (2) obtención de la plantilla QVT a partir del modelo del patrón, (3) se lleva a cabo la búsqueda del patrón y (4) las ocurrencias (submodelos) son devueltos.

MDPEM puede expresarse como una función f_{MDPEM} definida de la siguiente manera: $f_{MDPEM}(M_P, M_T) = Set(M_O)$, donde M_P representa al patrón de búsqueda, M_T representa el modelo origen sobre el que se realizará la búsqueda, y M_O que representa una ocurrencia de M_P sobre M_T . f_{MDPEM} puede devolver un conjunto de ocurrencias denominado $Set(M_O)$.

Tabla 2. Artefactos involucrados en el MDPEM

Nivel M2	Nivel M1	Descripción
PMM	PM	Los patrones (modelos) siguen un metamodelo dado, que debe ser compatible con el modelo donde se hará la búsqueda de ocurrencias.
QVTPMM	QVTPM	Los patrones se traducen a una representación en QVT (*Plantilla QVT* [33]), así, el metamodelo de plantilla QVT (QVTPMM) debe ser tomada en cuenta para generar el equivalente de un patrón dado en forma de plantilla QVT.
PIM / PSMMM	PIM/PSMM	Dado que cualquier patrón en un proceso MDA es un PIM o un PSM (según el nivel de abstracción), se supone que el modelo donde se realiza la búsqueda será un PIM o un PSM
-	Matching	After MDPEM is applied, matchings may be found. These matchings can be seen as fragments of the target model, so the metamodel for these elements can also be the PIM/PSMMM Tras la aplicación del MDPEM, las ocurrencias pueden considerarse como fragmentos del modelo donde se ha realizado la búsqueda, por lo que el metamodelo de las mismas también puede ser el PIMMM o el PSMMM
PMM = Pattern Metamodel; PM = Pattern Model; QVTPMM = QVT Pattern Metamodel; QVTPM = QVT Pattern Model; PIM/PSMMM = PIM/PSM Metamodel; PIM/PSMM = PIM/PSM Model		

En este paso, el concepto de MDPEM se extiende para ofrecer una funcionalidad adicional: los patrones (M_p) se ofrecen junto con la descripción de operaciones genéricas. Teniendo esto en cuenta, MDPEM se puede redefinir de la siguiente forma: en lugar de un patrón M_P, ahora MDPEM toma en su lugar un par <*patrón, operación genérica*>. Algebraicamente, esto se puede representar como $K = (M_T, A_A)$, donde M_T representa el mencionado patrón, y A_A que representa una operación genérica. De este modo f_{MDPEM} quedaría a su vez redefinido de la siguiente manera: $f_{MDPEM}(K, M_T) = Set(A_A(M_O))$. A_A se aplica a cada uno de las ocurrencias del conjunto M_O. De esta forma la operación genérica se "materializa" para cada ocurrencia obtenida en el MDPEM.

3.3. Generación del PIM

El PSM representa el esquema de la base de datos, pero empleando una terminología SQL-92 que queda muy lejos de la representación de un Servicio Web por la distancia intelectual es demasiado grande [36]. Sin embargo, empleando el PIM como representación la separación entre el PSM y el metamodelo PSM del WSDL se reduce notablemente, así como la complejidad de las transformaciones [37].

```
transformation SQLSchemaToOOSv2
   (sql92db: SQL92Schema, oos2: OOSv2){
key Class{name, owner};
key Association{name, owner};
key Property{name, owner};
top relation SQL92SchemaToOOSv2{…}
top relation TableToUMLElement{…}
top relation ConstraintToUMLElement{…}
relation ReferentialConstraintToAssociation{…}
relation UniqueConstraintToProperty{…}
relation BaseTableToClass{…}
relation ColumnToProperty{…}
relation DomainToUMLConstraint{…}
relation ViewToClass{…}
relation AssertionConstraintToPackageInvariant{…}
relation TableCheckConstraintToUMLConstraint{…}
//Functions
function SQL92_ValueToOOSv2V_Value
   (domain sql92 col:Column{}):
    domain oos2 val:ValueSpecification{}
function SQL92_TypeToOOSv2_Type
   (domain sql92 type: DataType{}):
    domain oos2 type:DataType{}
function DomConstraintToUMLClassInvariant
   (domain sqp92 cons:Constraint{}):
    domain oos2 cons:Constraint{}
   }
```

Figura 4. Transformación QVT para la obtención del PIM a partir del PSM

Así pues, previo a la generación de los Servicios Web, se obtiene una representación intermedia de la base de datos a partir del PSM: el PIM. El metamodelo del PIM está constituido por el metamodelo de UML 2 para la representación estática de sistemas [13]. Por este motivo, en el PIM los distintos elementos de la base de datos se representan mediante clases, asociaciones, propiedades, etc.

Para generar el PIM a partir del PSM se ha desarrollado una transformación QVT (Figura 4). Esta transformación está dividida en varios fragmentos o *relations*. En la terminología QVT, una *relation* es una porción de la transformación total donde se especifican un conjunto de condiciones que deben ser satisfechas por los modelos de entrada y salida. Por este motivo, cuando un elemento en el modelo destino no existe QVT fuerza su existencia. De este modo, cuando la transformación es ejecutada, las diversas *relations* son invocadas y el modelo destino (PIM) se va construyendo paso a paso. El algoritmo de transformación implementado en la transformación propuesta también detecta relaciones de herencia, actualizando el PIM para reflejar este tipo de relaciones.

Sin embargo, el PIM generado resulta pobre, ya que solamente contiene clases, propiedades, relaciones y asociaciones. Por esto, se considera una etapa posterior de reestructuración para mejorar el PIM con distintos tipos de operaciones, como las operaciones de acceso y las CRUD.

3.3.1. Adición de operaciones get/set. Las operaciones *get/set* son especificadas como modelos OCL. En realidad, existen modelos genéricos en OCL que se consideran como plantillas a la hora de instanciarlas en las correspondientes clases del PIM.

Siguiendo esta estrategia, se ha desarrollado una transformación QVT (Figura 5) para incluir estas operaciones en el PIM. Esta transformación se denomina "*in-place transformation"* en la terminología QVT, ya que el modelo de entrada y el de salida es el mismo, con lo que la t ransformación realmente produce una actualización del modelo.

```
transformation CreateGetSetOperations(pim:OOSv2){
   top relation createGetSetOperations{...}
   relation addSetOperation{…}
   relation createPostconditionForSetOperation{…}
   relation addGetOperation{…}
   relation createPostconditionForGetOperation{…}
}
```

Figura 5. Transformación QVT para la obtención del PIM a partir del PSM

3.3.2. Adición de operaciones CRUD. La persistencia de los objetos es un aspecto muy importante en los sistemas orientados a objetos. Siguiendo una estrategia similar a la planteada en el apartado 3.3.1, es posible actualizar el PIM dotándolo de un conjunto de operaciones CRUD [38] que nos permita gestionar la persistencia de las clases.

Sin embargo, las operaciones CRUD no se crean en las correspondientes clases del PIM, sino en una clase adicional: la *CRUDFacade* o fachada CRUD (Figura 1).

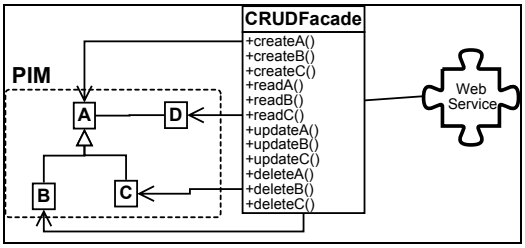


Figura 6. CRUDFacade con todas las operaciones del PIM

Las operaciones CRUD se sitúan en la *CRUDFacade*, ya que estas operaciones pueden considerarse como servicios útiles desde el punto de vista de cualquier otra aplicación en el contexto SOA que pudiera necesitar la funcionalidad que ofrecen este tipo de operaciones.

La Figura 7 muestra la actualización "*in-place*" desarrollada para la actualización del PIM con las operaciones CRUD. Esta transformación fuerza la creación de la clase *CRUDFacade* y dentro de ésta, por cada clase del PIM, las correspondientes operaciones CRUD.

```
Transformation addCRUDOperations (oos:OOSv2){
 top relation createCRUDOperations{...}
 relation createCreateOperation{...}
 relation createDeleteOperation{...}
 relation createReadOperation{...}
 relation createUpdateOperation{...}
 relation createPostconditionForCRUDOperation{...}
 relation createPostconditionForUpdateOperation{...}
 relation CreateAndBody{...}
}
```

Figura 7. Transformación QVT para la actualización del PIM con las operaciones CRUD

3.3. Descubrimiento de servicios en el PIM

Las bases de datos relacionales representan entidades del mundo real, y la mayoría de esas entidades están representadas explícitamente en el esquema de la base de datos en forma de tablas. Sin embargo, en algunas ocasiones esas entidades están representadas implícitamente mediante un conjunto de tablas [39]. En nuestro contexto, es posible extrapolar desde las bases de datos a la representación orientada a objetos (es decir, el PIM). Este tipo de clases "implícitas" puede ser considerado como *objetos abstractos* ya que pueden ser inferidos aunque realmente no existan.

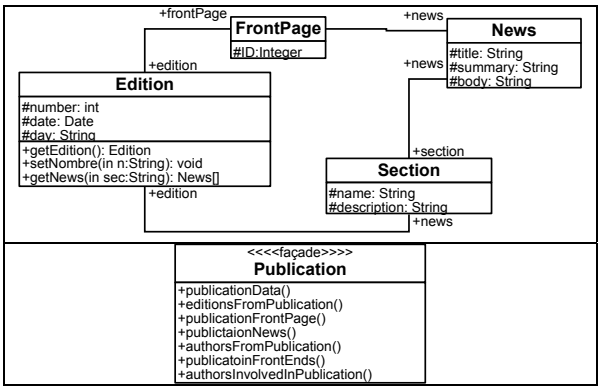


Figura 8. (arriba) Ejemplo de PIM y (abajo) un objeto abstracto.

Observando el PIM de la Figura 8 (arriba), es posible distinguir cuatro objetos diferentes: *Edition, FrontPage, News* y *Section*. Individualmente, dichas clases representan un concepto bien definido, sin embargo, al considerarlas todas juntas se puede deducir un objeto que no está explícitamente representado en el modelo: la *Publicación* (Figura 8 (abajo)).

En este sentido, la clase *Publicación* podría contener un conjunto de operaciones o servicios que involucraran las clases del PIM. Así, la clase *Publicación* jugaría también un papel de fachada para las funcionalidades que involucran al conjunto de clases que implícitamente representan al objeto abstracto.

Para el descubrimiento y la creación de esos objetos abstractos, el proceso propuesto define los siguientes pasos:

1. Identificar el conjunto de clases que componen el *objeto abstracto* (Figura 9).
2. Identificación del objeto abstracto y actualización el PIM con el mismo (Figura 10 (arriba)).
3. Identificación de servicios del objeto abstracto y actualización del PIM.

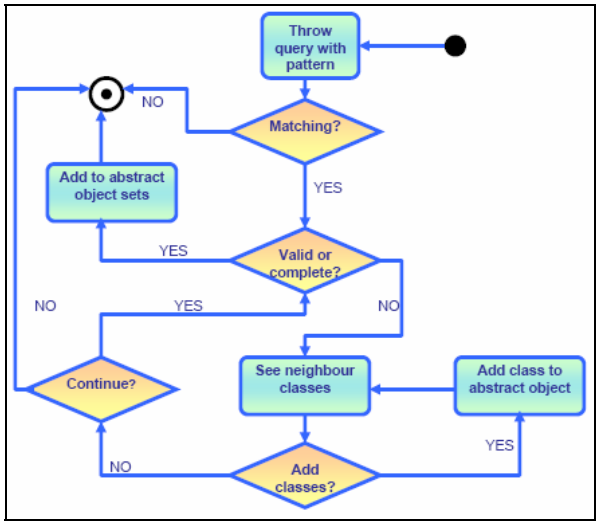


Figura 9. Algoritmo para la construcción del conjunto de clases componiendo el *objeto abstracto*

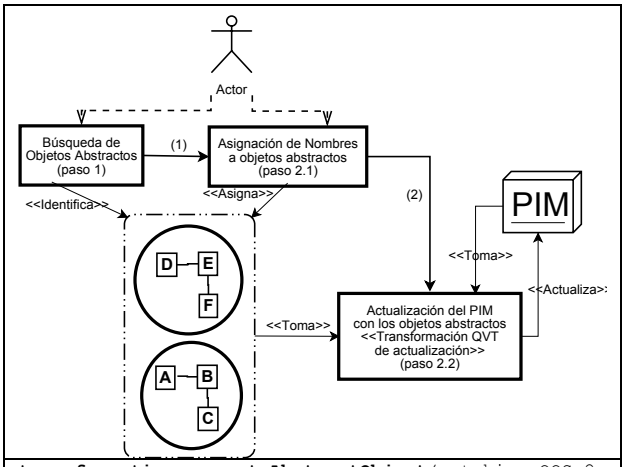
Para llevara cabo el primer paso se ha definido el algoritmo de la Figura 9. El algoritmo puede tomar como entrada tanto una única clase como un conjunto de clases a modo de patrón (siguiendo la idea propuesta por las "*Vistas Jerárquicas*" de [40]). La salida de este algoritmo, que debe ser supervisado por un actor humano, es un conjunto de clases que implementará el *objeto abstracto*.

Cuando se han definido los objetos abstractos del PIM, hay que asignarles nombre y crearlos en el PIM (es decir, actualizarlo). Para ello, en primer lugar se crean las clases en el PIM representando estos conceptos, y en segundo lugar se crean las relaciones necesarias entre las clases existentes y el nuevo objeto abstracto.

La Figura 10 (arriba) muestra cómo el ingeniero selecciona el conjunto de clases abstractas que conforma el objeto abstracto (obtenido de la aplicación del algoritmo de la Figura 9), y cómo el nuevo objeto abstracto es incluido en el PIM mediante una transformación QVT (Figura 10 (abajo)). Para esta transformación QVT, el modelo de entrada es el conjunto de clases que implementan el objeto abstracto y el nombre que se le asigna, mientras que la salida es el propio PIM con el nuevo objeto y sus correspondientes relaciones.

3.5. Generación de las interfaces WSDL

Tal y como se ha mencionado, este proceso sigue el estándar MDA (concretamente su evolución ADM), por lo que todos los artefactos involucrados son modelos. Incluso los Servicios Web son representados como modelos. En este proceso, se considera un metamodelo de WSDL (*Web Service Description Language*). La Figura 11 muestra una vista general de dicho metamodelo, desarrollado para este proceso.




```
transformation createAbstractObject(matching:OOSv2,
PIM:OOSv2)
{
top relation addAbstractObject{
  checkonly domain matching p:Package{name = pName}
  enforce domain PIM abs_obj:OOSv2{
     name = <Assigned_name>,
     isAbstract = 'FALSE'
  }
  where{
    p.ownedMember→forAll(c:Class |
      createAssociation(c, abs_obj);
  }
relation createAssociation{
  checkonly domain matching c:Class{
    name=cName
  }
  checkonly comain PIM abs_obj:Class{
    name=aOName,
    ownedAttribute = PROPS:Set(Property)
  }
  enforce domain assoc:Association{
    name=aName
    memberEnd = MEND: OrderedSet (Property),
    ownedEnd = OEND: OrderedSet (Property),
    navigableOwnedEnd = NOEND: OrderedSet (Property)
  }
  where{
    let c_Prop:Property = c.name;
    MEND = MEND->including (abs_obj);
    MEND = MEND->including (c);
    PROPS.including(c_Prop)
  }
}
}
```

Figura 10. (arriba) Asignación del nombre del objeto abstracto y (abajo) transformación QVT correspondiente.

En este paso, todas las clases fachada (*CRUDFacade*, objetos abstractos, etc) del PIM se transforman en modelos WSDL. Al generar las interfaces WSDL a partir de las clases fachada se consigue: (1) ocultar las complejidades del sistema representado por el PIM, y (2) elegir qué funcionalidades serán expuestas como servicios y cuáles quedarán ocultas.

Previo a la generación de modelos WSDL, el ingeniero debe seleccionar qué fachadas serán transformadas en este proceso. Es posible que algunas fachadas (como la *CRUDFacade*) contengan demasiadas operaciones, por lo que el ingeniero podrá "configurar" las fachadas bien separándolas en varias fachadas más pequeñas o, por el contrario, juntando varias fachadas para formar una sola de mayor

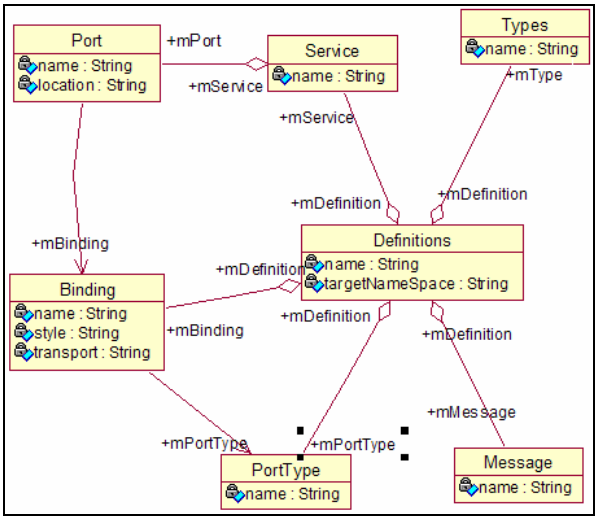


Figura 11. Fragmento del metamodelo de WSDL

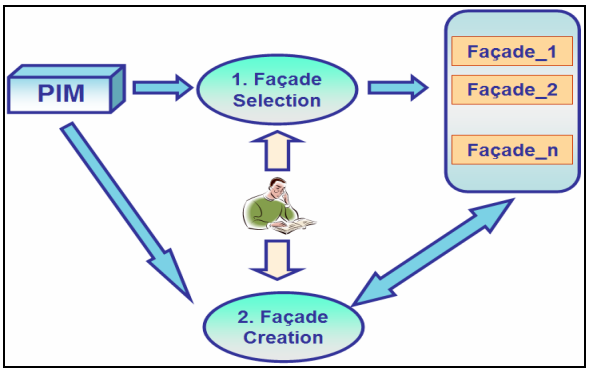


Figura 12. Elección y creación de fachadas para exponerlas como Servicios Web.

tamaño. Si por ejemplo, un conjunto de fachadas contuviera los servicios destinados a una aplicación cliente, éstas se podrían agrupar en una sola. Por el contrario, si algunos de los servicios de una fachada devolvieran datos privados que no deben ser accesibles para una determinada aplicación cliente, sería posible dividir la fachada en una que contuviera operaciones públicas y otra que contuviera operaciones privadas. La Figura 12 resume cómo el ingeniero puede manipular el conjunto inicial de fachadas para personalizar la fase de generación de modelos WSDL.

Una vez que el conjunto de fachadas ha sido constituido, una transformación QVT se encarga de generar automáticamente los modelos WSDL. La Figura 13 muestra las distintas *relations* de la que se compone la transformación *OOSv2ToWSDL*.

Esta transformación genera un conjunto de modelos WSDL a partir de las fachadas seleccionadas/creadas en el PIM, pero no el código fuente en sí del Servicio Web. Para esto último sería necesario utilizar o implementar algún tipo de generador de código fuente, para obtener la implementación de los Servicios Web a partir de los modelos WSDL.

```
transformation OOSv2ToWSDL(PIM:OOSv2, PSM:WSLayer)
{
  key Definitions (name);
  key PortType (name, mDefinition);
  key Binding (name, mDefinition);
  key Service (name, mDefinition);
  key Port (name, mService);
  key Message(name, mDefinition);
  key PortTypeOperation (name, mPortType);
  key BindingOperation (name. mBinding);

  top relation PackageToWSLayer{...}
  relation FacadeToWSDL{...}
  relation UMLOperationToService{...}
}
```

Figura 13. Transformación QVT para la generación de modelos WSDL a partir del PIM

4. Caso de estudio

Para verificar el proceso propuesto, se ha llevado un caso de estudio con una base de datos industrial. Los patrones presentados se han aplicado sobre la base de datos. Más adelante se mostrarán las ocurrencias (o *matchings*) obtenidos, así como algunos servicios interesantes.

4.1. Contexto

La aplicación industrial de la cual se ha obtenido la base de datos se denomina *Farolas*. Esta aplicación se desarrolló como marco de prueba de la arquitectura distribuida "*W10*". Esta aplicación ha sido cedida por la factoría de software Indra.

Farolas está compuesto de un cliente Web ligero y una base de datos relacional desplegada en un gestor Oracle. A continuación se muestran algunas de las características técnicas de la base de datos: 93 tablas, 2 vistas, 0 procedimientos almacenados, 105 índices, 234 restricciones (86 claves primarias, 106 check, 51 claves ajenas, y 0 claves *unique*).

4.2. Resultados

En esta sección se muestran los resultados de la aplicación del MDPEM con los patrones presentados en el apartado 3.2.1. Los patrones son lanzados contra el PSM que representa el esquema de la base de datos, obteniéndose un conjunto de ocurrencias de los mismos. Las tablas 4 a 7 muestran algunos de las ocurrencias obtenidas, mientras que la Tabla 3 resume los resultados totales de la aplicación del MDPEM.

La Tabla 3 muestra las ocurrencias que se han obtenido por cada uno de los patrones, así como el número de servicios derivados de los mismos. La columna de servicios descartados representa los servicios que han sido descartados por no ser útiles a priori. Tras entrevistar a algunos miembros del equipo de mantenimiento sobre el significado de algunas tablas, se nos reveló que las tablas nombradas con el patrón *"testX"* (por ejemplo, "*text5*" o "*test18*") no

contenían información relevante, y que su única función era la de mantener información de carácter temporal. Por este motivo, todas las ocurrencias de la búsqueda de patrones que involucrara alguna de estas tablas se ha descartado, y los servicios derivados se han considerado como "rechazados", ya que si en principio la información de las tablas no era relevante, tampoco lo serían los servicios vinculados con las mismas.

Tabla 3. Resultados de la aplicación del MDPEM para la primera extracción de servicios.

PATRÓN	#MATCHING	#SERVICIOS DERIVADOS	#SERVICIOS RECHAZADOS
FK	51	74	28
DFK	13	24	9
NFK	1	1	0
NFK2ONE	10	27	0

4.3. Discusión de los resultados

La aplicación *Farolas* consisten un conjunto de ficheros Java de código compilado y archivos de base de datos (archivos *dmp*). Sin embargo, dada su inexistencia no se suministró ningún tipo de documentación relacionada con el proceso de desarrollo o con la descripción de las tablas de la base de datos.

A pesar de que mediante entrevistas se obtuvo cierta información del personal de mantenimiento, ésta fue limitada, ya que no formaban parte del equipo original de desarrollo. La experiencia aportada sirvió para reconocer algunos servicios conocidos de entre los derivados de las ocurrencias obtenidas del MDPEM:

- *getContrato_Por_Factura(f:Factura)*: Empleando el patrón *FK* se deriva este servicio mediante el cual se obtienen los datos de un contrato a partir de una factura. La operación toma la factura como argumento devolviendo los datos del contrato.
- *getTipoConsumidor_Por_Sector(s:Secor)*: Empleando un patrón *DFK*, se obtiene este servicio que devuelve el tipo de un cliente dado un sector determinado, que se toma como argumento de entrada.
- *getContrato_Por_Factura_Y_Suministro(f: Factura, s:Suministro)*: Este servicio, obtenido de una ocurrencia del patrón *NFK2ONE*, toma una factura y un suministro, para devolver un contrato vinculado.

Tras las distintas interacciones que se realizaron con el personal de la empresa se concluyo que, aunque no todos los servicios obtenidos formaban parte de la aplicación original (bien porque no formaban parte realmente, o porque ellos no podían reconocerlos), es muy posible que esos servicios pudieran resultar útiles para desarrollos futuros en los que se viera involucrada la base de datos en estudio.

5. Conclusiones y trabajo futuro

Este artículo resume un proceso que sigue la filosofía ADM (*Architecture-Driven Modernization*). Este proceso esta desarrollado para la extracción de servicios a partir de bases de datos relacionales, de forma que estas puedan ser integradas en un contexto SOA.

Este proceso involucra un conjunto de metamodelos que restringen y representan todos los modelos generados a lo largo del proceso. La mayoría de las operaciones (principalmente las transformaciones, actualizaciones y la búsqueda de patrones) que involucran modelos han sido representadas mediante el lenguaje transformacional QVT.

El presente proceso ha sido validado mediante un caso de estudio realizado sobre una aplicación industrial. Los resultados obtenidos revelan que no sólo se han descubierto servicios reales existentes en la aplicación real, sino que además se han extraído una gran cantidad de servicios que podrían ser útiles en futuros desarrollos.

Surgen diversas líneas futuras a partir de este trabajo, como por ejemplo el descubrir más patrones para la aplicación del MDPEM. Una forma de obtener nuevos patrones será mediante la realización de más casos de estudio. Un ejemplo de patrones encontrados en la experimentación es el del patrón *NFK2ONE*, que se descubrió durante la realización del caso de estudio de la aplicación *Farolas* y que sirvió para generar un número considerable de servicios útiles.

6. Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto ENIGMAS (PBI-05-058), el proyecto ESFINGE (TIN2006-15175-C05-05/), y el proyecto FAMOSO (2007: FIT-340000-2007-71).

Tabla 4. Ocurrencias obtenidas tras la aplicación del patrón FK (fragmento)

A	B
EVO_SUMINISTRO	EVO_LECTURA
EVO_SUMINISTRO	EVO_TARIFICACION
EVO_FRANJA	EVO_TARIFICACION
EVO_CLIENTE	EVO_CONTRATO
EVO_CONTRATO	EVO_SUMINISTRO
EVO_CONTRATO	EVO_FACTURA
EVO_TARIFA	EVO_SUMINISTRO
EVO_TARIFA	EVO_FRANJA
GO_OPL_INTERFAZ	GO_OPL_ERROR
GO_SISTEMA	GO_OPL_ENLACE
GO_SISTEMA	GO_OPL_FUNCION
GO_OPL_FUNCION	GO_OPL_INTERFAZ
GO_OPL_FUNCION	GO_OPL_LOG
GO_OPL_TIPOS_FUNCION	GO_OPL_FUNCION

Tabla 5. Ocurrencias obtenidas tras la aplicación del patrón DFK (fragmento)

A	M	B
ARQW10_TEST_MAESTRO_A	ARQW10_TEST_PERF	ARQW10_TEST_MAESTRO_N
EVO_SUMINISTRO	EVO_TARIFICACION	EVO-FRANJA
EVO_CONTRATO	EVO_SUMINISTRO	EVO_TARIFA
GCCOM_NON_METERED_EQUIPMENT	GCCOM_CONCEPT_EQUIPMENT	GCCOM_CONCEPTS_MASTER
GCCOM_SECTOR	GCCOM_CONCEPTS_MASTER	GCCOM_PRORATE_TYPE
GCCOM_CONSUM_TYPE	GCCOM_EQUIPMENT_TYPE	GCCOM_SECTOR
XXX_PERFIL	XXX_ACCIONES_PERFIL	XXX_ACCIONES
XXX_ACCIONES	XXX_ACCIONES_METODOS_NEGOCIO	XXX_METODOS_NEGOCIO

Tabla 6. Ocurrencias obtenidas tras la aplicación del patrón NFK con N = 3

T1	T2	T3	M
GCCOM_EQUIPMENT_TYPE	GCCOM_EQUIPMENT_CALC_TYPE	GCCOM_NON_METERED_USAGES_DEF	GCCOM_NON_METERED_EQUIPMENT

Tabla 7. Ocurrencias obtenidas tras la aplicación del patrón NFK2ONE con N = 2

T1	T2	M
EVO_LECTURA	EVO_TARIFICACION	EVO_SUMINISTRO
EVO_SUMINISTRO	EVO_FACTURA	EVO_CONTRATO
EVO_SUMINISTRO	EVO_FRANJA	EVO_TARIFA
GCCOM_CONCEPTS_MASTER	GCCOM_EQUIPMENT_TYPE	GCCOM_SECTOR
GCCOM_NON_METERED_EQUIPMENT	GCCOM_NON_METERED_USAGES	GCCOM_NON_METERED_USAGES_DEF
GCCOM_NON_METERED_EQUIPMENT	GCCOM_NON_METERED_USAGES_DEF	GCCOM_EQUIPMENT_CACL_TYPE
GO_OPL_ENLACE	GO_OPL_FUNCION	GO_SISTEMA
GO_OPL_LOG	GO_OPL_INTERFAZ	GO_OPL_FUNCION
XXX_USUARIO	XXX_ACCIONES_PERFIL	XXX_PERFIL
XXX_ACCIONES_PERFIL	XXX_ACCIONES_METODOS_NEGOCIO	XXX_ACCIONES

7. Referencias

[1] Di Lucca, G.A., A.R. Fasolino, and P. Tramontana, *Reverse engineering Web applications: the WARE approach.* Journal of Software Maintenance and Evolution: Research and Practice, 2004. **16**: p. 71-101.

[2] Bézivin, J., S. Hammoudi, D. Lopes, and F. Jouault. *Applying MDA Approach for Web Service Platform*. in *In Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International (EDOC'04)*. 2004. Monterey (California, USA): IEEE Computer Society.

[3] Endrel, M., J. Ang, J. Arsanjani, S. Chua, P. Comte, P. Krogdahi, M. Luo, and T. Newling, *Patterns: Service-Oriented Architecture and Web Services. IBM - WebSphere Software*. 2004. p. 370.

[4] Colosimo, M., A. De Lucia, R. Francese, G. Scanniello, and G. Tortora. *MELIS:an Eclipse Based Environment for the Migration of Legacy Systems to the Web*. in *In Proceedings of the 13th Working Conference on Reverse Engineering (WCRE'06)*. 2006. Benevento, Italy: IEEE Computer Society.

[5] Li, S. and L. Tahvildari. *A Service-Oriented Componentization Framework for Java Software Systems*. in *In Proceedings of the 13th Working Conference on Reverse Engineering (WCRE'06)*. 2006. Benevento, Italy: IEEE Computer Society.

[6] Thiran, P., J.-L. Hainaut, and G.-J. Houben. *Database Wrappers Development: Towards Automatic Generations.* in *Euromicro Working Conference on Software Maintenance and Reengineering (CSMR´05)*. 2005.

[7] OMG, *MDA Guide Version 1.0.1.* 2003, Object Management Group. p. 62.

[8] OMG, *ADM Task Force*. 2006, Object Management Group.

[9] OMG, *Architecture-Driven Modernization Roadmap*. 2006, Object Management Group.

[10] ISO/IEC, *ISO/IEC 9075:1992, Database Language SQL*. 1992.

[11] Blaha, M. *A Retrospective on Industrial Database Reverse Engineering Projects-Part 1*. in *In Proceedings of the 8th Working Conference on Reverse Engineering (WCRE´01)*. 2001. Suttgart, Germany: IEEE Computer Society.

[12] Blaha, M. *A Retrospective on Industrial Database Reverse Engineering Projects-Part 2*. in *In Proceedings of the 8th Working Conference on Reverse Engineering (WCRE´01)*. 2001. Suttgart, Germany: IEEE Computer Society.

[13] OMG, *Unified Modeling Language: Superstructure. Versión 2.0*. 2005.

[14] OMG, *OCL 2.0 Specification. Version 2.0*. 2005, Object Management Group (OMG). p. 185.

[15] Cohen, Y. and Y.A. Feldman, *Automatic High-Quality Reengineering of Database Programs by Abstraction, Transformation and Reimplementation.* ACM Transactions on Software Engineering and Methodology, 2003. **12**(3): p. 285-316.

[16] Blaha, M. *Dimensions of Database Reverse Engineering*. in *Fourth Working Conference on Reverse Engineering (WCRE '97)*. 1997. Amsterdam, The NETHERLANDS.

[17] Henrard, J. and J.-L. Hainaut. *Data dependency elicitation in database reverse engineering*. in *Fifth European Conference on Software Maintenance and Reengineering (CSMR´01)*. 2001. Lisbon, Portugal: IEEE Computer Society.

[18] Sadiq, W. and M.E. Orlowska. *On Capturing Process Requirements of Workflow Based Business Information Systems*. in *Third International Conference on Business Information Systems (BIS'99)*. 1999. Poznan (Polonia): Springer-Verlag.

[19] Soutou, C., *Relational database reverse engineering: algorithms to extract cardinality constraints.* Data & Knowledge Engineering, Elsevier Science Publishers B. V., 1998. **28**(2): p. 161-207.

[20] Sousa, P.M.A., M.d.L. Pedro-de-Jesus, G. Pereira, and F. Brito e Abreu. *Clustering Relations into Abstract ER Schemas for Database Reverse Engineering*. in *Proceedings of the Third European Conference on Software Maintenance and Reengineering*. 1999. Amsterdam, Netherlands: IEEE Computer Society.

[21] Hainaut, J.-L., J. Henrard, J.M. Hick, D. Roland, and V. Englebert. *Database Design Recovery*. in *Eighth Conferences on Advance Information Systems Engineering*. 1996. Berlin.

[22] Henrard, J., V. Englebert, J.M. Hick, D. Roland, and J.L. Hainaut. *Program understanding in database reverse engineering*. in *Proceedings of the International Workshop on Program Comprenhension*. 1998.

[23] Hick, J.-M. and J.-L. Hainaut, *Strategy for Database Application Evolution: The DB-MAIN Approach.* LNCS 2813, 2003: p. 291-306.

[24] Thiran and J.-L. Hainaut. *Wrapper Development for Legacy Data Reuse*. in *Eighth Working Conference on Reverse Engineering (WCRE'01)*. 2001. Suttgart, Alamania: IEEE Computer Society.

[25] Bychkov, Y. and J.H. Jahnke. *Interactive Migration of Legacy Databases to Net-Centric Technologies*. in *Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE´01)*. 2001: IEEE Computer Society.

[26] García-Rodríguez de Guzmán, I., M. Polo, and M. Piattini. *An Integrated Environment for Reengineering*. in *Proceedings of the 21st International Conference on Software Maintenance (ICSM 2005)*. 2005. Hungary, Budapest: IEEE Computer Society.

[27] OMG, *Architecture-Driven Modernization Scenarios*. 2006, Object Management Group.

[28] OMG, *Architecture-driven Modernization (ADM): Knowledge Discovery Metamodel (KDM) Specification*. 2006, Object Management Group.

[29] van den Heuvel, W.-J. *Matching and Adaptation: Core Techniques for MDA-(ADM)-driven Integration of new Business Applications with Wrapped Legacy Systems*. in *Model-Driven Evolution of Legacy Systems (MELS 2004)*. 2004. Monterey, California, USA.

[30] Estévez, A., G.J. D., J. Padrón, and C. López. *System Integration Methodology Based on MDA*. in *European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA 2006)*. 2006. Bilbao (Spain): Springer-Verlag Berlin Heidelberg.

[31] García-Rodríguez de Guzmán, I., M. Polo, and M. Piattini. *A Framework for Model-Driven Pattern Matching*. in *Proceedings of the 9th International*

Conference on Enterprise Information Systems. 2007. Funchal, Madeira - Portugal (In press).

[32] García-Rodríguez de Guzmán, I., M. Polo, and M. Piattini. *Using Model-Driven Pattern Matching to Derive Functionalities in Models*. in *Proceedings of the Nineteenth International Conference on Software Engineering and Knowledge Engineering*. 2007. Boston, USA.

[33] OMG, *MOF QVT Final Adopted Specification*. 2005, Object Management Group.

[34] García-Rodríguez de Guzmán, I., M. Polo, and M. Piattini. *A Methodology for Database Reengineering to Web Services*. in *ECMDA-FA 2006*. 2006. Bilbao (Spain): Springer-Verlag Berlin Heidelberg.

[35] Calero, C., F. Ruiz, A. Baroni, F. Brito e Abreu, and M. Piattini, *An Ontological Approach To Describe the SQL:2003 Object-Relational Features.* Accepted in "Computer Standards and Interfaces", 2005: p. 28.

[36] Sommerville, I., *Software Engineering. 6th Edition*. 2000.

[37] Bézivin, J., S. Hammoudi, D. Lopes, and F. Jouault. *Applying MDA Approach to B2B Applications: A Road Map*. in *ECOOP Workshop on Model Driven Development (WMDD)*. 2004.

[38] Yoder, J. *Patterns for making business objects persistent in a relational database*. in *Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. 2002. Tampa Bay (Florida).

[39] Smith, J.M. and D.C.P. Smith, *Database abstractions: aggregation and generalization.* ACM Trans. Database Syst., 1977. **2**(2): p. 105-133.

[40] Keller, W. and J. Coldewey. *Relational Database Access Layers: A Pattern Language*. in *Proceedings of the (PloP 1996)*. 1996.