UN MÉTODO PARA LA CREACIÓN DE SERVICIOS WEB A PARTIR DE BASES DE DATOS RELACIONALES

Ignacio García-Rodriguez de Guzmán, Macario Polo, Mario Piattini

Grupo de Investigación ALARCOS
Departamento de Sistemas de Información y Tecnologías
Instituto de Investigación y Desarrollo UCLM-Soluziona
Universidad de Castilla-La Mancha
Paseo de la Universidad, 4 – 13071 Ciudad Real, España
{Ignacio.GRodriguez, Macario.Polo, Mario.Piattini}@uclm.es

Palabras clave: Reingeniería, metamodelo, Servicio Web, QVT, patrones, MDA

Resumen. *Actualmente, las bases de datos son uno de los elementos más importantes de los sistemas de información, almacenando toda su información. Aunque han surgido nuevos estándares, la mayoría de las bases de datos siguen siendo SQL-92. Siguiendo la filosofía SOA, actualmente se tiende a exponer servicios hacia la Web, haciéndolos disponibles tanto para otros usuarios como para el propio sistema. Tratar con sistemas heredados suele ser muy complejo, por ello en este artículo se propone una metodología que, siguiendo un enfoque MDA, implementa un proceso de reingeniería para descubrir servicios en bases de datos SQL-92 y exponerlos como Servicios Web.*

1. INTRODUCCIÓN

Los sistemas de información están compuestos por muchos elementos, entre los que podemos encontrar documentación, programas, hardware, bases de datos, etc. De todos ellos, las bases de datos pueden ser consideradas como la piedra angular del sistema, puesto que almacenan la mayor parte de los datos que el sistema gestiona. A pesar de que se han desarrollado múltiples versiones del estándar SQL (como SQL-99 y SQL:2003), muchos sistemas están funcionando todavía con bases de datos relacionales [1, 2] basadas en el estándar SQL-92 [3], por lo que éstas cobran gran importancia.

Los programas que utilizan bases de datos heredadas son en ocasiones programas heredados con una baja mantenibilidad, lo que produce un esfuerzo excesivo cuando se trata con el. Cualquier intento por actualizar este tipo de sistemas heredados resulta complejo por razones como el tamaño y la complejidad [4], falta de experiencia en el lenguaje, falta de documentación e incluso el miedo a cambiar algo que funciona, aunque mal desarrollado.

Sin embargo, existen técnicas como la reingeniería que resulta muy útil para tratar con este tipo de sistemas. Últimamente, la ingeniería inversa ha sido relacionada estrechamente con la arquitectura dirigida por modelos [5]. En pocas palabras, MDA permite separar la lógica de negocio de la plataforma de implementación del sistema [5], trabajando a nivel de modelo y

metamodelo, y junto con la ingeniería inversa forman lo que se conoce como ADM (Architecture-Driven Modernization) [6], que integra la ingeniería inversa y MDA. Algunos metamodelos, como CWM [7] han sido estandarizados para soportar sistemas heredados [8], siendo vitales durante la ingeniería inversa.

Siguiendo la filosofía MDA y ADM se está desarrollando una metodología centrada en las bases de datos, mediante la que se implementa un proceso donde se emplean técnicas de reingeniería. El PSM inicial es una base de datos SQL-92; los distintos PIMs están constituidos por el conjunto de metamodelos utilizados durante todo el proceso, mientras el PSM final está constituido por el conjunto de Servicios Web a generar.

El desarrollo de un proceso de reingeniería parcialmente automatizado requiere especificar qué tipo de base de datos relacional va a ser analizada. A pesar de que el actual estándar SQL es SQL:2003, la mayoría de las bases de datos actuales están definidas bajo el estándar SQL-92. Por esto, la metodología se centra en esta versión del estándar.

Este artículo está estructurado de la siguiente manera: la Sección 2 resume algunos trabajos relacionados con el contexto de este artículo; en la Sección 3, se presenta una visión global del proceso; en la Sección 4, se detalla cómo se genera el modelo conceptual; en la Sección 5 se citan ciertas mejoras que se realizan sobre este modelo; la Sección 6, resume la estrategia para la búsqueda de servicios; la Sección 7 trata sobre la implementación de dichos servicios en forma de Servicio Web y, la Sección 8, resume las conclusiones y el trabajo futuro.

2. ESTADO DEL ARTE

A diferencia de otras áreas de la ingeniería del software, la investigación en la reingeniería de datos no ha sido uno de los puntos en los que más se ha profundizado por dos sencillas razones: (1) la clásica separación de la ingeniería del software y los sistemas de bases de datos, y (2) la ingeniería inversa del código fuente parece ser más interesante en los entornos académicos [9]. Este reingeniería de datos puede llevarse a cabo para [10-12]: redocumentación, migración de modelos, reestructuración, mantenimiento, etc.

Se han propuesto muchos algoritmos para la recuperación de metadatos de las bases de datos, como en [13, 14], donde se analiza cómo extraer la estructura de bases de datos relacionales mediante algoritmos. J.L. Hainaut, lidera el proyecto DB-MAIN [15-17], centrado en varios aspectos de la reingeniería de bases de datos.

El entorno MIDAS [10] soporta la migración de bases de datos en red a relacionales, permitiendo además las subrutinas de acceso por código SQL. El *wrapping* de bases de datos también puede verse como otro tipo de migración, donde mediante el uso de adaptadores se conecta la base de datos con otros entornos. Estos adaptadores se encargarían de abstraer el modelo de datos de la base de datos, traduciendo las consultas en caso necesario [18]. Los adaptadores o *wrappers* permiten integrar la base de datos en un entorno distribuido [19], actuando como interfaces o *fachadas* entre la base de datos y el sistema externo.

En [20], se ofrece un enfoque distinto, donde los autores implementan un proceso (mediante una herramienta denominada *RelationalWeb*) en el que, utilizando técnicas de reingeniería, generan aplicaciones para la gestión de bases de datos relacionales.

Con mayor frecuencia, la investigación llevada a cabo sobre bases de datos se ha centrado en los temas previamente mencionados (migración, reestructuración, etc.); sin embargo según la documentación estudiada, no existen demasiados trabajos que se centren en la generación de servicios a partir de bases de datos relacionales, por lo que siguiendo la línea de [20], se investigará en este tema tratando además de ofrecer los servicios mediante Servicios Web.

3. VISIÓN GLOBAL DE LA METODOLOGÍA

La metodología que se propone toma como punto de inicio una base de datos basada en el estándar SQL-92. Esto se debe a que según algunos estudios [1, 2] realizados sobre la industria, muchos de los sistemas de información actuales siguen funcionando con bases de datos basadas en este estándar. Partiendo de este punto, los resultados pueden ser extendidos a posteriores versiones del SQL. La **Fig. 1** muestra los diferentes pasos de los que está compuesta la metodología. En el primer paso, se extrae la estructura de la base de datos mediante ingeniería inversa, produciendo un modelo que representa base de datos relacional bajo estudio.

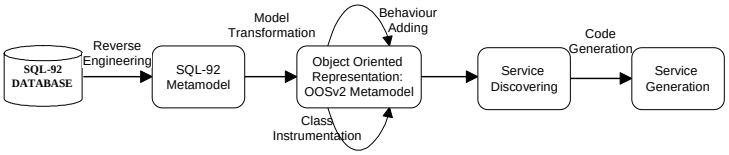

Fig. 1. Visión general del proceso.

En el siguiente paso, este modelo obtenido en el paso anterior se transforma en otro modelo que representa la base de datos mediante una notación orientada a objetos (descrito en términos de clases, asociaciones y propiedades).Tras esto, el ingeniero es el responsable de guiar el proceso de descubrimiento de servicios. El resultado es un conjunto de servicios representados de manera abstracta. La última fase es la de la generación de código.

4. OBTENCIÓN DEL MODELO CONCEPTUAL DE LA BASE DE DATOS

4.1. Extracción de metadatos para la representación del esquema

En [20, 21] se presenta una microarquitecutra eficaz para la recuperación del esquema de la base de datos. Todos los metadatos recuperados son almacenados y representados mediante el metamodelo (adaptado de [22]) de SQL-92 de la **Fig. 2**, al que se ha denominado *SQL92Schema*. En él, se consideran los elementos más importantes del estándar. A partir de ahora, nos referiremos a este metamodelo como SQL92Schema.

4.2. Transformación del metamodelo *SQL92Schema* en *OOSv2*

La instancia del metamodelo *SQL92Schema*, es transformada a una representación conceptual orientada a objetos. Para ello, se utiliza un subconjunto del metamodelo de UML 2 [14] (que, no obstante, no se muestra en ninguna figura por razones de espacio).

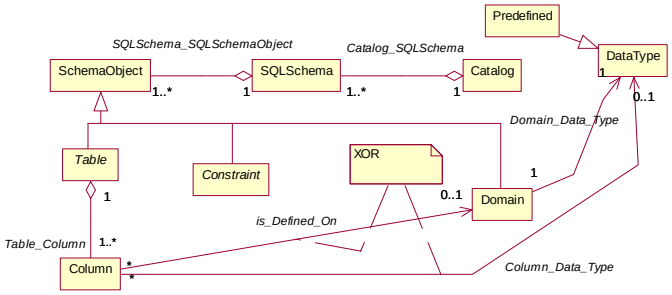

Fig. 2. Vista general del metamodelo de SQL-92

A continuación se explica cómo obtener una representación orientada a objetos a partir del esquema de una base de datos mediante transformación QVT compuesta. Dado que todos los elementos son modelos, el proceso de reingeniería es independiente de la plataforma en la que se trabaje, y el descubrimiento de servicios por lo tanto es realizado a nivel conceptual.

Tabla 1. Transformación para la obtención de un sistema orientado a objetos a partir de un esquema SQL-92

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** AssertToConstraint{…} **relation** TableCheckConstraintToUMLConstraint{…} //Funciones **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{}

El lenguaje que se eligió para llevar a cabo las transformaciones fue QVT [23]. QVT es un potente lenguaje que permite especificar transformaciones entre modelos y metamodelos. QVT incluye una notación sintáctica y gráfica para la definición de transformaciones.

Dada la extensión de la transformación completa, en la Tabla 1 sólo se muestra una pequeña parte del algoritmo de transformación, compuesto por las cabeceras de las relaciones que establecen cómo transformar los elementos de *SQL92Schema* en elementos de *OOSv2*.

El algoritmo es una función compuesta que invoca, entre otras, la relación mostrada en la **Fig. 3** (en notación gráfica). Todas las relaciones involucradas en la transformación son invocadas durante la ejecución de la transformación global *SQLSchemaToOOSv2(sqldb:SQL92Schema, oos2:OOSv2)* de la Tabla 1.

La relación de la **Fig. 3** tiene como objetivo tomar una columna de una tabla y transformarla en una propiedad de una clase.

5. INSTRUMENTACIÓN DE LAS CLASES

En esta etapa del proceso, todo el trabajo se centra en las instancias de los metamodelos *OOSv2* y *SQL92Schema*. Esto significa que todas las clases involucradas en un servicio son en realidad una representación orientada a objetos de las tablas de la base de datos, y los parámetros necesarios.

Antes de realizar ninguna tarea relacionada con la búsqueda de posibles servicios, se agregan operaciones básicas a las clases de la instancia de *OOSv2*. Estas operaciones añadidas automáticamente son operaciones *get* y *set* para todas las propiedades y las denominadas CRUD (*Create, Read, Update y Delete*)

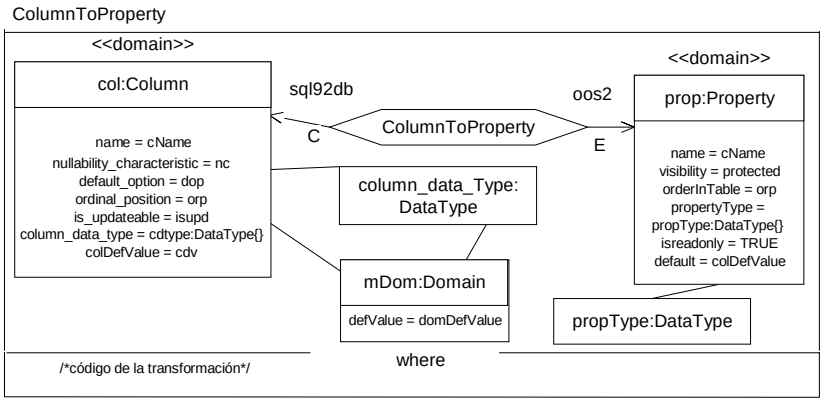

Fig. 3. Relación QVT para obtener una propiedad UML a partir de una columna SQL

Las clases también pueden anotarse con operaciones adicionales. Por ejemplo, imaginemos la clase *Cuenta*, generada a partir de una tabla *Cuenta* de la base de datos. Es muy probable que la clase *Cuenta* necesite operaciones como *ingresar*, *retirar* o *consultarSaldo*. Este tipo de operaciones pueden representarse mediante una máquina de estado vinculada a la clase. Los estados y las transiciones entre ellos pueden ser anotados manualmente por el ingeniero de software (véase [20] para una descripción detallada de esta actividad), o ser descubiertos a partir de los datos almacenados en la base de datos. Las transiciones entre los estados deben ser asignadas siempre manualmente por el ingeniero, ya que resulta casi imposible establecer de forma automática la relación entre éstos.La automatización del descubrimiento de estados en las clases se basa en el uso de heurísticas. Algunas de ellas son:

- *Uso de valores límite*: en las columnas numéricas, los valores límite pueden ser utilizados para establecer intervalos que puedan ser vistos como estados. Así, en el ejemplo de la tabla (y clase) *Cuenta*, la columna *Saldo* podría ser utilizada para realizar esta inferencia: *SaldoNegativo* ($-\infty < Saldo < 0$), *SaldoCero* ($Saldo = 0$) y *SaldoPositivo* ($0 < Saldo < \infty$).
- *Mediante las claves ajenas*: supongamos una tabla, y que una de sus columnas referencia mediante una clave ajena a otra tabla con una sola columna (que además posee solamente un conjunto restringido de valores, como si fuera una enumeración). Puede suponerse que esos valores estén definiendo los posibles estados en los que pueda encontrarse un registro de la primera tabla. Por ejemplo, dos tablas (Proyectos_Fin_Carrera y Estado_Proyecto), donde la primera representa proyectos fin de carrera (PFC) y la segunda acota los posibles estados por los que puede pasar un PFC (a saber, *propuesto, aprobado, asignado, suspendido, finalizado*). La columna *etapa* de la primera tabla referencia a la segunda mediante una clave ajena, de forma que para cada proyecto pudiera saberse cual es su etapa actual. Así, se podrían tomar los valores de la segunda tabla como posibles estados, generando una máquina de estados con estos estados.

6. EXTRACCIÓN DE SERVICIOS PARA BASES DE DATOS SQL-92

6.1. Representación de los servicios

Según [24], un servicio es una función bien definida, auto contenida y que no depende del contexto o estado de otros servicios.

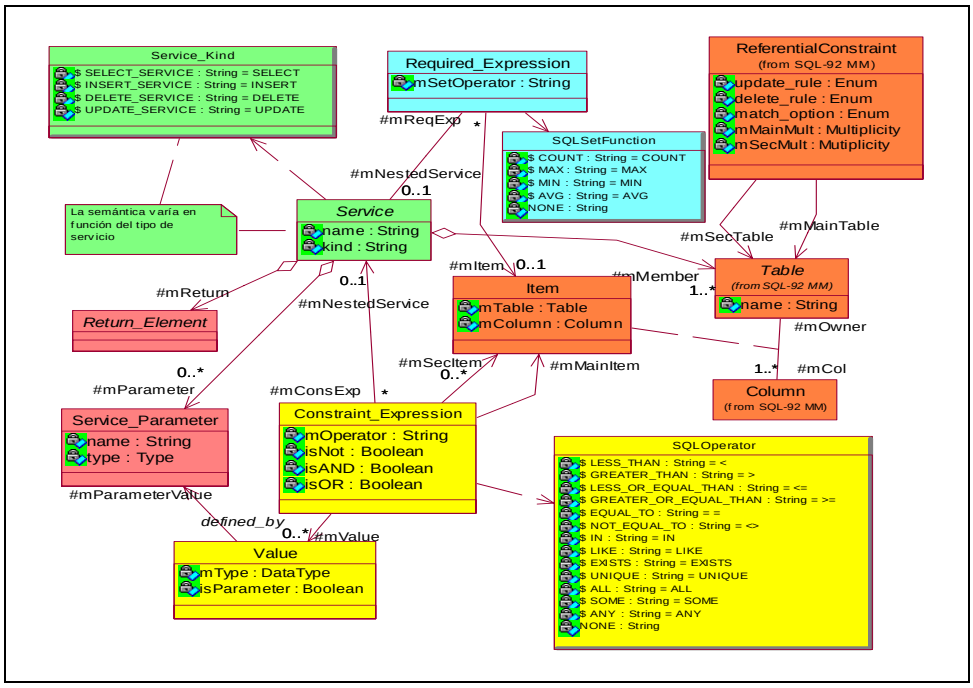

Fig. 4. Metamodelo de Servicio

Quizá uno de los temas a tener en cuenta antes de afrontar la tarea de generar servicios automáticamente es cómo construir o representar esos servicios de manera genérica. Dado que un metamodelo se puede ver como un lenguaje abstracto que especifica qué aspectos de una parcela del mundo real pretenden representarse [25], se ha desarrollado un metamodelo para poder representar cualquier servicio basado en una base de datos.

Un servicio puede dividirse en dos partes básicas: la componente estática y la componente dinámica. La componente estática representa todos los elementos involucrados en el servicio. Por otro lado, la componente dinámica de un servicio esté relacionada con el comportamiento del servicio, es decir, la secuencia de operaciones que deben realizarse para que el servicio sea ejecutado satisfactoriamente.

En la implementación final de cada servicio, cada uno de ellos será traducido a una (más o menos compleja) sentencia SQL: es decir, tanto los resultados del *Model-Driven Pattern Matching*, como de las operaciones CRUD y el resto de operaciones que puedan disparar transiciones entre estados de la máquina de estados que anota la clase.

Debido a la representación SQL de los servicios, no es necesaria una descripción o componente dinámica. Con este propósito, se ha desarrollado un metamodelo de servicio (Fig.

4), que aunque no pretende cubrir la sintaxis completa del estándar SQL-92, trata de abarcar lo necesario para esta metodología. En lugar de utilizar la notación BNF de SQL-92 se ha desarrollado este metamodelo debido a que (1) resulta mucho más fácil de manejar e integrar con MDA, (2) la notación de modelo resulta mucho más compatible con el resto de metamodelos empleados en todo el proceso y (3) este metamodelo contiene todos los elementos necesarios para generar el código fuente para implementar los servicios en la generación de código. Dada la versatilidad y complejidad del metamodelo, es necesario anotar algunas de sus clases con OCL, para evitar la generación de modelos incorrectos.

6.2. Model-Driven Pattern Matching

Mediante un ejemplo, se va a introducir el concepto de *Model-Driven Pattern Matching* (MDPEM en adelante), que tan importante resulta para esta metodología. Siendo M_A un modelo que contiene el conjunto de elementos que toman parte en un servicio, es posible buscar coincidencias de este modelo dentro de otro modelo mayor, llamado M_B, para encontrar ocurrencias del primero. Esto es, M_A sirve de patrón para encontrar ocurrencias dentro de M_B. M_B puede ser tanto la instancia del metamodelo *SQL92Schema* como la del metamodelo *OOSv2* (en otras palabras, el esquema relacional o el diagrama de clases obtenido). El objetivo de este proceso es la obtención de un conjunto con todos los *matchings* del patrón, para posteriormente elegir los que se ajusten a las necesidades.

Esta idea se corresponde con el concepto de(MDPEM. Esbozada en MDA [5], adquiere más importancia en QVT [23], dada la naturaleza del lenguaje. En este contexto, el patrón es una descripción genérica de un servicio que se utilizará para realizar el proceso de *matching* contra el modelo que represente la base de datos. A tal fin, se proponen los metamodelos de patrón de la **Fig. 5**. A diferencia de otros metamodelos [26], los propuestos en la **Fig. 5** (a y b) son mucho más sencillos, ya que están acotados en un contexto determinado.

Dado que el MDPEM puede llevarse a cabo a dos niveles distintos de abstracción (*SQL92Schema* y *OOSv2*), se ha desarrollado un metamodelo de patrón para cada uno de los niveles (**Fig. 5** (a) y (b) respectivamente).

7. IMPLEMENTACIÓN DEL SERVICIO

En la sección anterior se ha ido presentando una metodología para la reingeniería de bases de datos SQL-92 para el descubrimiento de servicios. El objetivo principal es el de desarrollar un método que ayude al ingeniero del software a exponer servicios basados en un sistema heredado, concretamente bases de datos SQL-92.

El método elegido para publicar los servicios es la tecnología de Servicios Web, ya que estos fueron concebidos para la integración, interoperación y encapsulación de sistemas heredados [27, 28]. Así pues, por un lado, los Servicios Web fueron elegidos porque el principal elemento de la metodología es un sistema heredado como lo son las bases de datos SQL-92 (aún muy extendidas, como ya se ha comentado) y porque los Servicios Web trabajan sobre cualquier plataforma y tecnología.

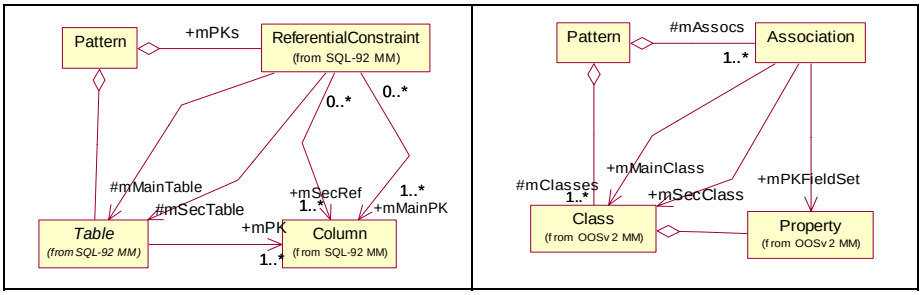

Fig. 5. Metamodelos de patrones

Actualmente, se están definiendo las transformaciones QVT para generar los Servicios Web a partir de las especificaciones abstractas de los servicios. Respecto a esta transformación, los elementos más importantes son los servicios abstractos y el documento WSDL (que puede verse como la interfaz del Servicios Web).

Los modelos origen para esta última transformación en la metodología son los servicios encontrados y los modelos que representan la base de datos; los modelos destino son el metamodelo del documento WSDL y la implementación de los servicios.

8. CONCLUSIONES Y TRABAJO FUTURO

Hasta ahora, se ha estado desarrollando una aplicación denominada Relational Web [20], para la reingeniería de bases de datos. Habiendo sido probada satisfactoriamente en varios proyectos, servirá como base para la implementación del soporte CASE para la metodología tratada en este artículo.

A pesar de la obsolescencia de SQL-92, muchos sistemas siguen trabajando sobre bases de datos basadas en este estándar, razón por la que se está desarrollando esta metodología.

En la sección 4.1, se ha presentado la parte correspondiente a SQL-92 del metamodelo, seleccionado para representar la base de datos recuperada. El metamodelo original está pensado para representar bases de datos SQL-2003, por lo que servirá como punto de extensión para futuras versiones de la metodología.

La idea principal es el descubrimiento de funcionalidades en la base de datos mediante búsqueda con patrones. Las funcionalidades extraídas se harán accesibles mediante Servicios Web, que son automáticamente generados como parte del proceso de reingeniería.

Las transformaciones utilizadas son descritas utilizando QVT, sin embargo no se ha encontrado software capaz de soportar el lenguaje completo, por lo que se han implementado mediante algoritmos.

9. AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por el proyecto MÁS (Mantenimiento Ágil del Software), Ministerio de Ciencia y Tecnología/FEDER, TIC2003-02737-C02-02; el proyecto ENIGMAS, Plan Regional de Investigación Científica, Desarrollo Tecnológico e Innovación, Junta de Comunidades de Castilla-La Mancha, PIB-05-058, el proyecto FAMOSO,

parcialmente financiado por el Ministerio de Industria, Turismo y Comercio, 2006: FIT-340000-2006-67 Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica 2004-2007 y el "Fondo Europeo de Desarrollo Regional (FEDER)", Unión Europea, y el proyecto MECENAS (Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia, PBI06-0024).

10. REFERENCIAS

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

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

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

[4]. Wang, X., et al. *Business Rules Extraction from Large Legacy Systems*. in *Proceedings of the Eighth Conference on Software Maintenance and Reengineering*. 2004. Tampere, Finland

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

[6]. Bézivin, J. *Model Engineering for Software Modernization*. in *Guest Talk in the 11th IEEE Working Conference of Reverse Engineering*. 2004

[7]. OMG. *Common Warehouse Metamodel (CWM) Specification*. 2003. <http://www.omg.org/docs/formal/03-03-02.pdf> [Consultado el: 29-09-2005]

[8]. Favre, J.-M., M. Godfrey, and A. Winter. *Integrating Reverse Engineering and Model Driven Engineering*. in *Proceedings of the Second International Workshop on Meta-Models and Schemas for Reverse Engineering (ATEM 2004)*. 2004. Delft, The Netherlands

[9]. Müller, H.A., et al. *Reverse Engineering: A Roadmap*. in *International Conference on Software Engineering - Proceedings of the Conference on The Future of Software Engineering*. 2000. Limerick, Ireland

[10]. 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.

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

[12]. 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 IEEE Computer Society

[13]. 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.

[14]. Sousa, P.M.A., et al. *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 IEEE Computer Society

[15]. Hainaut, J.-L., et al. *Database Design Recovery*. in *Eighth Conferences on Advance Information Systems Engineering*. 1996. Berlin

[16]. Henrard, J., et al. *Program understanding in database reverse engineering*. 2002

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

[18]. 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 IEEE Computer Society

[19]. 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 IEEE Computer Society

[20]. 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 IEEE Computer Society

[21]. Polo, M., et al., *Generating three-tier applications from relational databases: a formal and practical approach.* Information & Software Technology, 2002. **44**(15): p. 923-941.

[22]. Calero, C., et al., *An Ontological Approach To Describe the SQL:2003 Object-Relational Features.* Accepted in "Computer Standards and Interfaces", 2005: p. 28.

[23]. OMG. *MOF QVT Final Adopted Specification*. 2005. <http://www.omg.org/docs/ptc/05-11-01.pdf> [Accessed in February, 2005]

[24]. WSSOA. *Web Services and Service-Oriented Architectures*. 2005. <http://www.service-architecture.com> [Consultado el: 14-12-2005]

[25]. Bézivin, J. *MDA™ : From Hype to Hope, and Reality - Invited talk at UML´2003*. 2003

[26]. Pagel, B.-U. and M. Winter. *Towards Pattern-Based Tools*. in *Proceedings of EuropLop*. 1996

[27]. Alonso, G., et al., *Web Services. Concepts, Architectures and Applications*, ed. M.J. Carey and S. Ceri. 2004, Berlin: Springer. M.J. Carey and S. Ceri. p. 354.

[28]. Lavery, J., et al. *Laying the Foundations for Web Services over Legacy Systems*. in *Proceedings of the Fourth International Workshop on Web Site Evolution (WSE'02)*. 2002. Montreal, Canada