

GERENCIA TECNOLOGICA

INFORMATICA

Informatics Technology Management

Generación de Modelos de Requisitos

Mejora en la Gestión de Calidad
de la Información

El Factor Humano en el Proceso
de Mantenimiento de Software

Metamodelos en la Generación
automática de Aplicaciones Web

Técnica para la resolución de
problemas

Conceptos sobre Minería Web

Tutorías virtuales para los Nuevos
Modelos Educativos

Ambiente Virtual Adaptativo

ITI
Colombia
Instituto Tecnológico Colombiano
de Informática de Colombia

Universidad
Industrial de Santander
CIDLIS

Número 7, Volumen 3

**GERENCIA
TECNOLÓGICA**

INFORMÁTICA

Informatics Technology Management

DIRECTOR

Dr. Ricardo Llamosa Villalba

nrllamos@iticol.org

DISEÑO Y DIAGRAMACIÓN

Victor H. Cardenas

Jose F. Esquivia G.

IMPRESIÓN

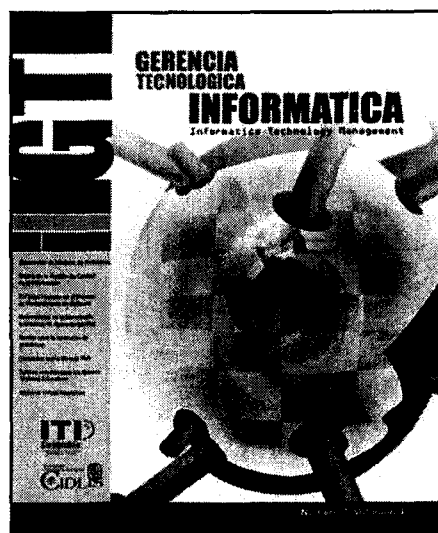
División Editorial y de Publicaciones UIS

Universidad Industrial de Santander

Carrera 27 Calle 9a Ciudad Universitaria

Tel. 6348418

publicac@uis.edu.co



Una Publicación de **ITI COLOMBIA**

Para el sector de la investigación, académico y empresarial

revistagti@iticol.org

Número 7, Volumen 3. 13-12-2004

Publicidad y Comercialización

ITI Colombia

Cra 19 No. 35 - 02

Of. 330 - Tel. 6520123

Bucaramanga - Colombia

ISSN 1657-8236

La dirección de la revista GTI no se responsabiliza por el contenido de los artículos ni su publicación en otros medios

Arbitraje

Miguel ángel Gómez-Nieto
Universidad de Córdoba
Córdoba, España

Luís Enrique Carreño
Universidad Industrial de Santander
Bucaramanga, Colombia

Mariela Curiel
Universidad Simón Bolívar
Caracas, Venezuela

Carlos Chávez Rodríguez
Pontificia Universidad Católica del Perú
Lima, Perú

Abraham Dávila
Pontificia Universidad Católica del Perú
Lima, Perú

Ismael Caballero Muñoz-Reja
Universidad de Castilla-La Mancha
Ciudad real, España

Alicia Martínez Rebollar
Universidad Politécnica de Valencia
Valencia, España

Hugo Estrada
Universidad Politécnica de Valencia
Valencia, España

Ricardo Llamosa Villalba
Universidad Industrial de Santander
Bucaramanga, Colombia

Adriana Llamosa Ardila
adriana@cidlisuis.org

Martha de Florez
martha@cidlisuis.org

Ricardo Llamosa Villalba
nrllamos@cidlisuis.org

**GRUPO DE INVESTIGACIÓN
TESIS-CIDLIS**
Tecnologías y Estándares
en Ingeniería del Software

Comité
Editorial

Comité de Redacción

Victor Hugo Cárdenas
vcardenas@cidlisuis.org

Juddy Alexandra Gómez
Juddy@cidlisuis.org

Ruth Claudia Moreno
ruclamo@cidlisuis.org

Andrés Guerrero
anguerrco@cidlisuis.org

Victor Hugo Sánchez
vsanchez@cidlisuis.org

Luis Enrique Carreño
luisenca@cidlisuis.org

Óscar Miranda
osanim@cidlisuis.org

Edwin Logreira
elogreira@cidlisuis.org

Luz Elena Gutiérrez
luzelena@cidlisuis.org

Marlon José Mujica
mujica@cidlisuis.org

Lilia Castellanos
lilia@cidlisuis.org

Sandro Castellanos
sances@cidlisuis.org

**UNIVERSIDAD INDUSTRIAL
DE SANTANDER**

Alvaro Beltrán Pinzón
Rector

Lucila Niño Bautista
Vicerrectora Académica

Emérita Oliveros de Rivas
Vicerrectora Administrativa

Adolfo León Arenas
Decano Facultad de Físico - Mecánicas

Ricardo Llamosa Villalba
Director Científico CIDLIS

CIDLIS

Centro de Innovación y Desarrollo
para la investigación en Ingeniería del
Software

Adriana Llamosa Ardila
adriana@cidlisuis.org
Líder Grupo CISMA

Ricardo Llamosa Villalba
nrllamos@cidlisuis.org
Líder Grupo GUIA

Juddy Alexandra Gómez
juddy@cidliuis.org
Líder Grupo TESIS

Martha Liliana Suarez
msuarez@cidlisuis.org
Líder Grupo GAITA

TESIS

Tecnologías y Estándares
en Ingeniería del Software

**Apoyo
Institucional**

Índice

EDITORIAL	10
GENERACIÓN DE MODELOS DE REQUISITOS A PARTIR DE MODELOS ORGANIZACIONALES: UN ENFOQUE BASADO EN PATRONES	11
Alicia Martínez Rebolgar, Hugo Estrada Esquivel, Óscar Pastor López	
MEJORA EN LA GESTIÓN DE CALIDAD DE LA INFORMACIÓN	23
Ismael Caballero, Miguel Ángel Blanco, Mario Piattini	
EL FACTOR HUMANO EN EL PROCESO DE MANTENIMIENTO DE SOFTWARE	33
Élvira Rolón, Francisco Ruiz, Mario Piattini	
UNA HERRAMIENTA BASADA EN METAMODELOS PARA LA GENERACIÓN AUTOMÁTICA DE APLICACIONES	49
Ignacio García Rodríguez de Guzmán, Macario Polo, Mario Piattini	
UNA TÉCNICA BASADA EN COMPONENTES PARA LA RESOLUCIÓN DISTRIBUIDA DE PROBLEMAS	59
Luis Iribarne	



CONCEPTOS SOBRE MINERÍA WEB **71**
José Aguilar, Altamiranda Junior

**SISTEMA DE GESTIÓN DE TUTORÍAS
VIRTUALES COMO SOPORTE DE LOS
NUEVOS MODELOS EDUCATIVOS** **79**
Miguel Ángel Gómez, Gonzalo Cerruela García,
Irene Luque Ruiz, Francisco Calvillo Muñoz

**AMBIENTE VIRTUAL ADAPTATIVO DE
GESTIÓN DE CONTENIDOS DE ENTRENAMIENTO** **87**
Ricardo Llamosa Villalba, Andres Guerrero, Juddy Gómez,
Ruth Moreno, Victor Sanchez, Oscar Miranda,
Lilia castellanos, Luis Carreño



Editorial

Se concluye un año más de la existencia de nuestra revista Gerencia Tecnológica Informática / Informatics Technology Management y con ello se persiste en un empeño complejo de permanencia y esfuerzo en beneficio del sector de la tecnología informática.

En esta oportunidad se cubre la temática de la Ingeniería del Software en aspectos de factores humanos, educación, gestión de calidad, requisitos, sistemas adaptativos, y modelos de sistemas y minería de datos en la Web.

Este número constituye el fin del ciclo de vida necesario para realizar la solicitud de indexación de la revista ante COLCIENCIAS*. Por esta razón es preciso agradecer a todos los que han contribuido en la permanencia periódica de la revista en los dos últimos años.

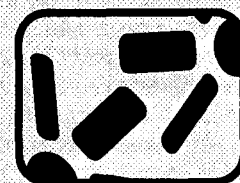
Para concluir este mensaje es imprescindible insistir en mi invitación para que los investigadores nacionales o internacionales participen con sus contribuciones con lo cual nuestro esfuerzo se verá recompensando al lograr producir los próximos números en el año 2005.

Muchas gracias, de nuevo, un saludo muy cordial de la familia ITI y CIDLIS.

* Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología

Dr. Ricardo LLamosa Villalba
Director Científico ITI Colombia
Director Científico CIDLIS - UIS

Una herramienta basada en metamodelos para la generación automática de aplicaciones web



IGNACIO GARCÍA-RODRÍGUEZ DE GUZMÁN

Doctor (C) Grupo de Investigación
ALARCOS, Universidad de Castilla-La
Mancha
España
PAIS (ESPAÑA)

MACARIO POLO

Docente, Grupo de Investigación
ALARCOS, Universidad de Castilla-La
Mancha
España
PAIS (ESPAÑA)

MARIO PIATTINI

Director, Grupo de Investigación
ALARCOS, Universidad de Castilla-La
Mancha
España
PAIS (ESPAÑA)

RESUMEN.

En este artículo, se describe la arquitectura de una herramienta que genera aplicaciones web de tres capas a partir de bases de datos relacionales. La herramienta está dotada de un conjunto de metamodelos que nos permiten desarrollar el proceso completo de reingeniería. Mediante el acceso al diccionario de la base de datos, la herramienta crea una instancia del metamodelo de la base de datos aplicando una técnica de ingeniería inversa; a partir de esta instancia y, mediante un proceso de reestructuración, se obtiene una instancia de un metamodelo de clases que nos permite transformar el esquema relacional en un diagrama de clases. Por último y según las necesidades del usuario, a través de un proceso de ingeniería directa, la herramienta genera una nueva instancia del metamodelo de clases para poder así representar la aplicación final. A nivel conceptual, se define un conjunto de funciones para implementar las transformaciones mencionadas; dichas funciones son implementadas por la herramienta mediante un conjunto de fábricas. Los metamodelos aquí citados son el núcleo de nuestra aplicación.

PALABRAS CLAVES

Metamodelo, Reingeniería, Ingeniería Inversa, Aplicación Web, Base de Datos, EJB

1. INTRODUCCIÓN

Según [2], la reingeniería es el proceso de aplicar, en primer lugar, ingeniería inversa a un sistema existente con el fin de obtener un conjunto de estructuras de mayor nivel de abstracción que lo representen para, en una fase posterior, aplicar ingeniería directa con el fin de construir una nueva versión del sistema original. Este nuevo sistema probablemente estará dotado de nuevas funcionalidades o simplemente será un software de mayor calidad. Es por esto

que para la primera fase, se debe obtener un conjunto de especificaciones abstractas a partir del sistema heredado, que son usadas a posteriori, para construir una nueva implementación del sistema software (Figura 1).

Aunque sea lo más frecuente, el código fuente no es siempre el punto de inicio del proceso de reingeniería. En realidad, muchos autores han estudiado y propuesto técnicas para la aplicación de la ingeniería inversa en otro tipo de productos como las bases de datos. En este caso, el objetivo era obtener un modelo conceptual que represente el dominio del problema original a modo de diagrama entidad-relación [1][4][5] o de diagrama de clases [8]. El uso de diagramas de

clase en lugar de esquemas entidad-relación supone, desde un punto de vista de reingeniería, la posibilidad de aprovechar las ventajas que nos ofrece las características del paradigma de la orientación a objetos, ya no sólo para la posterior reconstrucción de la base de datos, sino también para la implementación de aplicaciones capaces de manipular la base de datos.

Este artículo describe los metamodelos incluidos en una herramienta que permite generar aplicaciones web de tres capas completas a partir de bases de datos relacionales, y se muestra también cómo estos metamodelos son integrados en la arquitectura de la herramienta. Todas las bases de datos relacionales comparten el mismo conjunto de elementos principales como las tablas, las claves primarias, las claves ajenas o los disparadores, siendo posible por ello, representarlas a todas mediante un metamodelo independiente del fabricante. En algunas ocasiones, cada fabricante utiliza un conjunto de tipos de datos diferentes, y por esto, la forma de recuperar la estructura de la base de datos puede variar según el gestor de bases de datos. A fin de dotar a la herramienta de la capacidad de instanciar el metamodelo de base de datos para productos de diferentes fabricantes, se ha definido una fábrica abstracta con tantas implementaciones concretas como tipos de bases de datos queramos tratar. Una vez que tenemos el metamodelo de la base de datos, mediante la aplicación de una nueva función obtenemos el metamodelo de un diagrama de clases.

Con esta instancia de este nuevo metamodelo estamos representando la capa de negocio de la aplicación Web que nos proponemos generar. Este último metamodelo es transformado una vez más en otro nuevo metamodelo, esta vez dependiente de la plataforma destino en la que la aplicación web será generada. Este última instancia sigue representado a la capa de negocio de la aplicación final. La

información contenida en este último metamodelo es imprescindible para, (1) generar las capas de presentación y persistencia de la aplicación final, y (2) la fase de generación del código completo de la aplicación.

Este artículo está organizado de la siguiente manera: la Sección 2 ofrece una vista general sobre algunas técnicas de ingeniería inversa relacionadas con nuestro estudio; la Sección 3 es el núcleo del artículo, describiendo los metamodelos y la arquitectura de la herramienta de generación; finalmente, en la Sección 4 mostramos nuestras conclusiones y líneas de trabajo futuro.

2. ESTADO DEL ARTE

La mayoría de los trabajos de ingeniería inversa que tratan sobre las bases de datos, están centrados en la recuperación de los esquemas conceptuales utilizados en el desarrollo inicial de las mismas, a fin de poder migrar la base de datos antigua (Codasyl, por ejemplo) a otros modelos más modernos, o cambiar el sistema de gestión de la base de datos, o la detección de errores de integridad, etc. En nuestro caso, pretendemos obtener aplicaciones web que permitan la gestión de sus datos.

Distintos autores han realizado diferentes propuestas para la ingeniería inversa de bases de datos relacionales. Hainaut et al. [5] proponen un método general para aplicar ingeniería inversa a cualquier sistema de bases de datos o colección de ficheros de datos. Este método consiste en cuatro grandes procesos, exactamente para ir de la ingeniería inversa hacia la ingeniería directa: (1) Extracción de las Estructuras de datos; (2) Conceptualización de las Estructuras de Datos; (3) Unstranslation, que detecta construcciones dependientes respecto al sistema de gestión de bases de datos; y (4) Normalización Conceptual, que recupera las estructuras de alto nivel producidas en la fase de ingeniería directa.

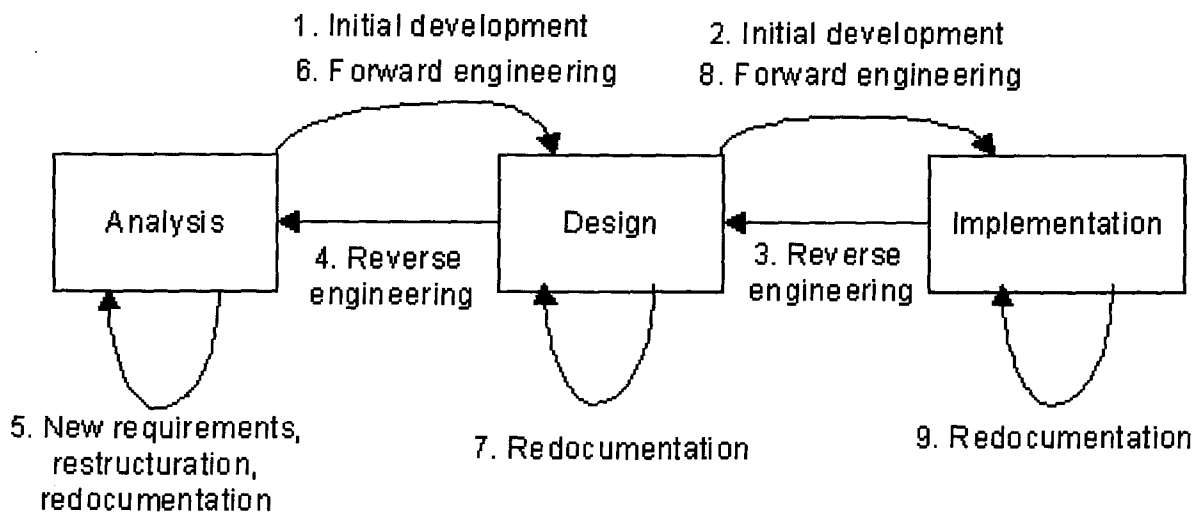


Figura 1. Implementación de Sistema Software

Pedro de Jesús y Sousa [7] presentan un interesante estudio en el que analizan las características de distintas propuestas (estado inicial de la base de datos, artefactos obtenidos, etc) y proponen un método que permite el uso de todas ellas. Se analizan ocho métodos de distintos autores, salvo uno, que obtiene diagramas de clases en OMT. El resto de métodos produce esquemas conceptuales ER o ER-extendidos. Estos métodos generan los mejores resultados cuando los datos heredados cumplen un conjunto de precondiciones deseables, como estar en al menos la 3ª Forma Normal, falta de inconsistencias, etc. El método de Pedro de Jesús y Sousa aprovecha todas las ventajas de los anteriores, ya que obtiene un conjunto de clusters de bases de datos, cada uno de los cuales contiene elementos de acuerdo a su afinidad para realizar la ingeniería inversa con uno de los métodos que analizan. Al final, su "macrométodo" produce un esquema entidad-relación que es utilizado como punto de inicio para el proceso de reconstrucción de la base de datos. Andersson [1] extrae el posible esquema entidad-relación utilizado inicialmente para construir la base de datos mediante el análisis de lenguajes de manipulación de datos, buscando en cada una de las sentencias SQL embebidas en el código fuente de los programas que usan la base de datos. De esta forma, por ejemplo, las claves primarias y las ajenas son identificadas a través del estudio de las condiciones de unión existentes en las sentencias SQL.

De la misma forma que los usuarios gestionan la información contenida en una base de datos mediante un conjunto de programas externos a la misma, la estructura de los documentos en los que la base de datos está representada (modelos de datos), tiene una gran influencia en los correspondientes documentos que representan los programas (modelos funcionales). El paradigma orientado a objetos incrementa la cohesión de ambos modelos unificando los datos y el comportamiento bajo un mismo punto de vista. Así, el esquema entidad-relación obtenido de la etapa de ingeniería inversa puede ser traducido a un diagrama de clases para aprovechar las ventajas de las construcciones orientadas a objetos, las herramientas, etc. para la siguiente etapa de ingeniería directa. En [8], se describe un algoritmo para obtener un diagrama de clases a partir de una base de datos relacional; ahora hemos mejorado el metamodelo utilizado y su integración en una herramienta para generar aplicaciones.

3. ESTRUCTURA Y COMPORTAMIENTO DE LA HERRAMIENTA

Esta sección describe la estructura de la herramienta, explicando cómo funciona en cada fase del proceso de reingeniería.

3.1. INGENIERÍA INVERSA Y REESTRUCTURACIÓN

La Figura 2 muestra la pantalla principal de nuestra herramienta. De la misma manera que puede trabajar con cuatro gestores de bases de datos distintos (Oracle, Caché Intersystems, SQL Server y Microsoft Access), la herramienta posee cuatro conjuntos diferentes de formularios para conectar con ellas. Cuando se presiona cualquiera de los botones "Reverse", la herramienta intenta conectar con la base de datos deseada y, si la conexión es posible, instancia la

fábrica apropiada que lee el diccionario de datos y construye una instancia del metamodelo de la base de datos relacional.

Estas fábricas son especializaciones de una fábrica abstracta que incluye la operación getBD, la cual debe ser instanciada en cada una de las fábricas especializadas.

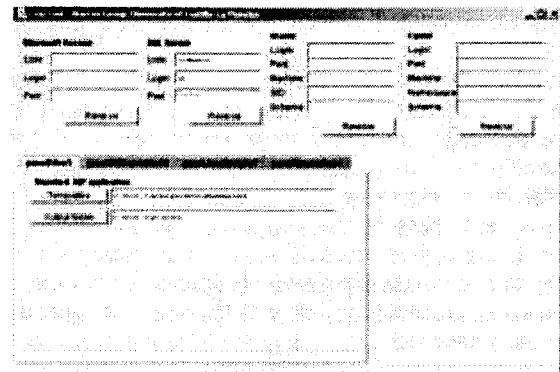


Figura 2. Pantalla principal de la aplicación

A pesar de que la aplicación está implementada en Java y que utiliza la API JDBC (que incluye un conjunto de interfaces que ofrecen la posibilidad de gestionar bases de datos de una manera uniforme), no todos los fabricantes implementan todas las operaciones de estas interfaces. Por ello, el código de cada fábrica tiene pequeñas diferencias con respecto a las otras. La Figura 4 describe la estructura del metamodelo utilizado para representar una base de datos relacional. En la versión actual, la herramienta considera la presencia de tablas, relaciones de integridad referencial entre tablas, columnas y procedimientos almacenados. Cada columna almacena información acerca de su nombre, si es parte o no de la clave primaria y si es parte o no de algún índice único. Además, almacena un valor entero que representa su tipo de dato (el campo mField), que depende del fabricante. Con respecto a esto último, por ejemplo Oracle y SQL Server asignan diferentes valores numéricos para el tipo VARCHAR. Posteriormente, en la fase de generación de código, se implementa algún tipo de mapeo a fin de mantener la correspondencia entre los tipos de datos del fabricante y los tipos de datos de la aplicación final.

Para preparar el proceso de generación de código, la instancia del metamodelo de base de datos es traducido a una instancia de otro metamodelo que representa un diagrama de clases (Figura 5): la clase OOS representa un Sistema Orientado a Objetos que es creado a partir de una base de datos sobre la que se ha aplicado ingeniería inversa. Cada OOS conoce a su conjunto de clases y relaciones; cada relación conoce a dos clases (la de la izquierda y la de la derecha, identificadas en la figura mediante los roles mLeft y mRight) y almacena información sobre la cardinalidad de cada uno de los lados de la relación (corresponde a los campos mCardL y mCardR) y sobre la navegabilidad (campos booleanos mFinalLeft y mFinalRight).

El último campo en la clase Relationship (mNumber) representa un número único dado a esta relación, que podría ser útil cuando una misma clase está relacionada con otra

clase mediante más de una relación.

La clase Class en la Figura 5 representa un calificador del modelo de clase, y almacena información sobre su naturaleza (si es interfaz o clase) y sobre su abstracción. La clase conoce también a las otras clases en el diagrama de clases que son superclases suyas (rol mSuperclasses), su conjunto de campos y sus métodos.

La herramienta construye un OOS a partir de una instancia de la base de datos. Inicialmente, agrega una clase por cada tabla, y por cada columna agrega un campo a la clase correspondiente. Las claves ajenas entre tablas son traducidas en asociaciones entre clases. Esas relaciones de clave ajena en la base de datos cuya multiplicidad en ambos sentidos es uno, son consideradas como relaciones de herencia, lo que puede implicar la eliminación de la base de datos de aquellos campos que sean definidos en la superclase. En este paso del proceso de reingeniería, la herramienta no agrega ningún método a la clase, ya que se realiza en la siguiente etapa.

Una clase adicional es agregada al sistema a fin de permitir la

ejecución de los procedimientos almacenados definidos en la base de datos. Se añade un método público en esta clase por cada uno de los procedimientos almacenados. El parámetro types debe coincidir con el conjunto de tipos de datos del lenguaje de programación de la aplicación final.

3.2. INGENIERÍA DIRECTA Y GENERACIÓN DE CÓDIGO

La versión actual de la aplicación es capaz de generar aplicaciones web de tres capas: la capa de presentación consiste en un conjunto de páginas JSP y HTML que permiten la interacción entre el usuario y los datos almacenados en la base de datos. Todas las operaciones ejecutadas por el usuario son enviadas a la correspondiente clase en la capa de dominio, que tiene implementados los métodos CRUD [10]:

- Un constructor "vacío", que asigna a cada campo de la clase un valor por defecto, como el 0 para los enteros o el null para los strings.
- Un constructor materializador, que construye una instancia de la clase a partir de un registro de la base de datos.
- Los métodos de inserción, borrado y actualización, que

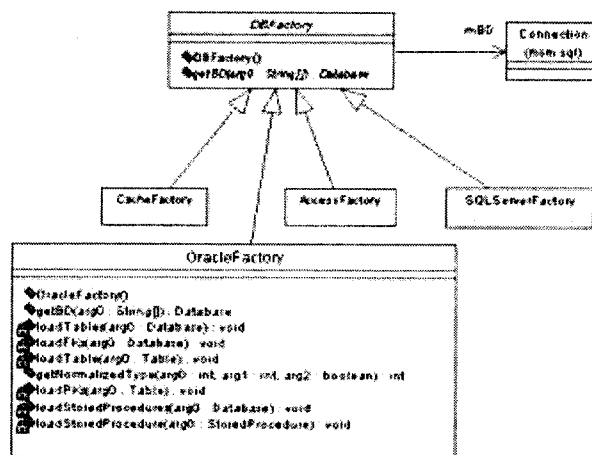


Figura 3. Conjunto de fábricas para tratar con las diferentes bases de datos

añaden nuevos registros a la base de datos, los eliminan y los actualizan.

También se añaden métodos get y set en cada clase, al igual que otras operaciones que permiten la navegación entre los registros relacionados.

La capa de persistencia tiene sólo una clase, la llamada Broker [3] de base de datos, que actúa como mediador entre la capa dominio y la de persistencia.

Posteriormente, y antes de la generación de código, la instancia del metamodelo de diagrama de clases es traducido a otra instancia de un metamodelo de clases dependiente de la

plataforma. En la capa de dominio de la aplicación final, la herramienta puede generar bien componentes Enterprise Javabeans (EJB) o clases Java estándar. Puede considerarse un EJB como un tipo especial de clase que no tiene constructores y que tiene asociada una "clase clave primaria". Este EJB es accesible para el cliente mediante dos interfaces que son gestionadas por un contenedor.

Cuando un cliente desea ejecutar cualquier tipo de operación en una instancia de un EJB, utiliza la llamada interfaz Home del EJB, que es expuesta mediante el contenedor; el contenedor localiza el componente y devuelve al cliente una referencia denominada interfaz Remote, que el cliente utiliza para

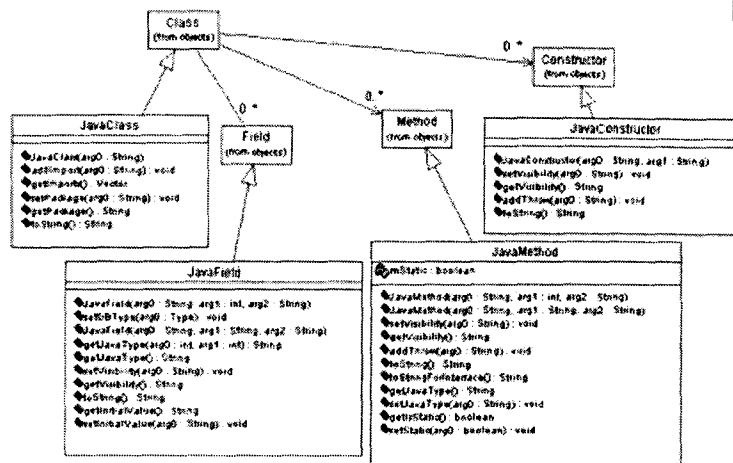


Figura 6. Consideraciones especiales para aplicaciones Java

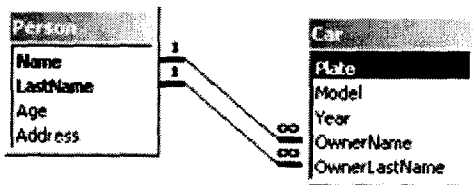


Figura 7. Una base de datos sencilla

Si la aplicación no va a estar basada en EJBs sino en clases Java estándar, dos clases son generadas para la capa de negocio: una clase de dominio pura que contiene las operaciones CRUD junto con los métodos get y set, y una clase adicional (patrón Fabricación Pura [6]) para recuperar resultados en formato HTML y permitir la navegabilidad entre las clases relacionadas (Figura 10). En la capa de presentación, por cada clase/tabla original, se crea un conjunto de páginas JSP para manipular los registros de la tabla. A diferencia de lo anterior, las páginas JSP no están representadas en ningún metamodelo, siendo generado de

acuerdo a un conjunto de plantillas.

De esta forma, una página JSP llamada listX se genera para recuperar listas de registros de la tabla X (lado superior izquierdo de la Figura 11), y una página formX para gestionar las instancias de la clase X. Estos formularios corresponden a tablas que referencian a otras tablas, utilizando botones para recuperar los valores de los campos correspondientes de la tabla referenciada, como se observa abajo en la Figura 11.

Dependiendo de si utilizamos EJB o clases Java estándar para la capa de dominio de la aplicación, existirán pequeñas diferencias entre el código de las páginas JSP generadas (Tabla 1).

El proceso general de reingeniería se describe en la Figura 12: una de las fabricas extrae la estructura de la base de datos obteniendo el objeto Database, que corresponde al metamodelo de base de datos (Figura 4). Después, la información del metamodelo de la base de datos es convertido en una instancia de metamodelo de diagrama de clases (Figura 5), que es la base para la generación de código por los distintos generadores de código fuente.

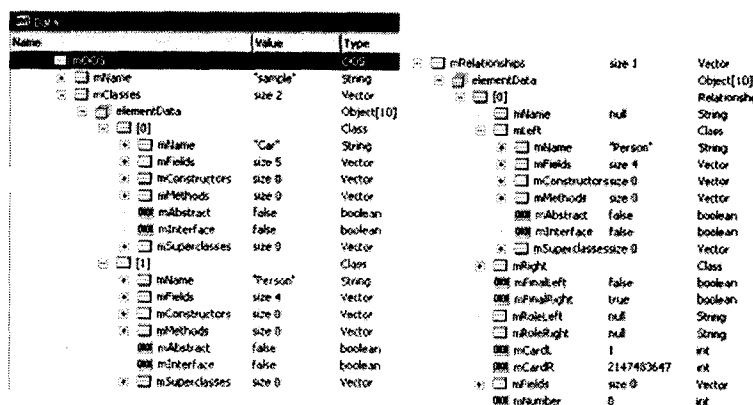


Figura 8. Valores de la instancia del metamodelo del diagrama de clases para la base de datos

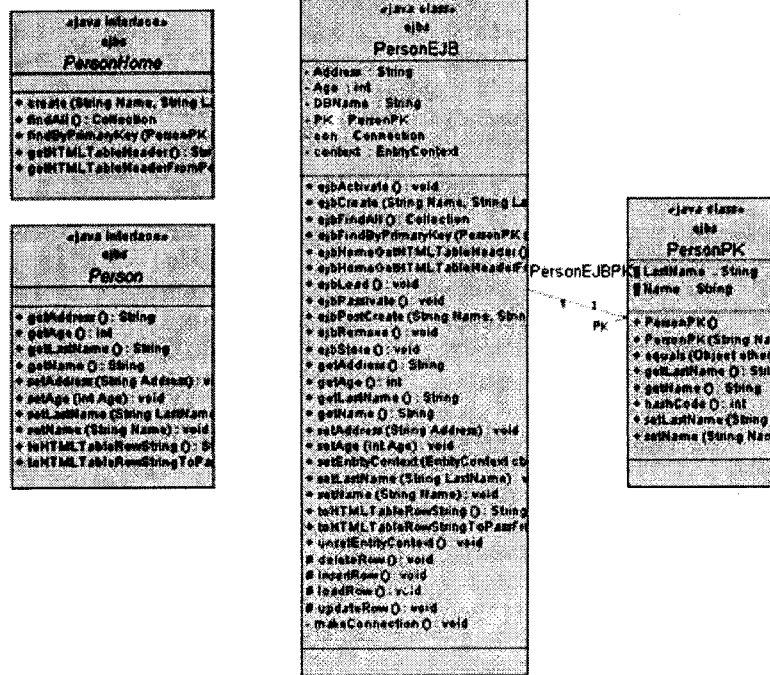


Figura 9. Cuando usamos EJBs, se generan diversas clases para la capa de negocio

```

<%= home.getHTMLTableHeader() %>
<%
    Collection col=home.findAll();
    Iterator it=col.iterator();
    while (it.hasNext()){
        Person ejb=(Person) it.next();
        out.print(ejb.toHTMLTableRowString());
    }
%>
<%= FPPerson.getHTMLTableHeader() %>
<% FPPerson.getList(bd, out, ""); %>
With EJBs
With Standard Java classes

```

4. CONCLUSIONES Y TRABAJO FUTURO

En este artículo se ha presentado detalles arquitectónicos de una herramienta para la generación de código a partir de bases de datos. La herramienta ya ha sido utilizada en algunos proyectos, en los que la adecuada distribución de responsabilidades a través de las capas ha permitido que, para un sistema de información de gestión habitual, tan sólo el 3% de las líneas de código hayan tenido que ser escritas a mano.

Hemos realizado un análisis de mantenibilidad del código fuente generado, lo que nos ha permitido la predicción de

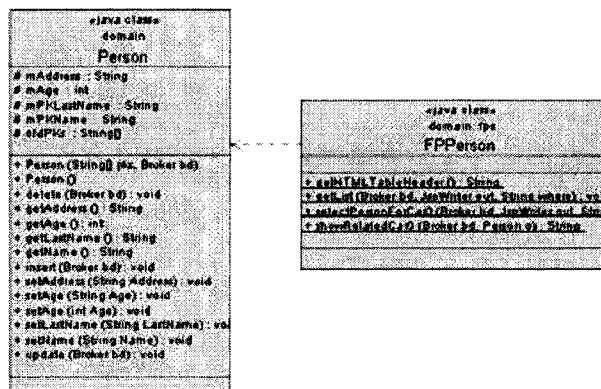


Figura 10. Algunas clases se generan para la capa de negocio cuando usamos clases Java estándar

algunas métricas del código anteriores a la generación de la aplicación web. Las siguientes ecuaciones nos ayudan a predecir métricas como LOC (Lines of Code), WMC (Weighted Methods per Class) y CBO (Coupling Between Objects) para cada uno de los EJB generados, cuando la opción para la generación de la aplicación web ha sido de EJBs (K_{LOC} y K_{WMC} con constantes que representan el mínimo tamaño y complejidad para cada componente). Por lo general, todos los valores están por debajo de los umbrales establecidos por [9].

La arquitectura de la aplicación es muy sencilla de extender: la implementación de una nueva fábrica para generar código a partir de la base de datos de otro fabricante, no tomaría más de una hora. En principio no hemos necesitado generar aplicaciones para otros entornos que no estén basados en Java, pero la implementación de nuevos generadores de código podría ser igualmente sencillo.

Actualmente, estamos definiendo un nuevo metamodelo para representar máquinas de estado, y poder así describir operaciones adicionales en las clases de dominio, que sean invocadas desde la capa de presentación. Para mejorar el funcionamiento de la herramienta, estamos desarrollando un entorno gráfico para representar el Sistema Orientado a

Objetos que obtenemos a partir de la base de datos, para así poder personalizar el conjunto de clases o EJBs obtenidos.

En la actualidad, la herramienta no genera adecuadamente código en la capa de presentación para tipos de datos definidos por el usuario, siendo esta una de nuestras líneas de trabajo futuro.

Las bases de datos relacionales son entornos complejos con muchos elementos; por ello, hemos planeado mejorar el metamodelo actual de base de datos para ofrecer soporte a todos los elementos de las bases de datos relacionales, además de su migración al paradigma objeto relacional. Este último punto, estimamos que no sea muy difícil de alcanzar debido a la naturaleza del sistema obtenido y de la arquitectura de la aplicación, basada en metamodelos y fábricas.

5. AGRADECIMIENTOS

Este trabajo es parte del proyecto MÁS (Mantenimiento Ágil del Software), Ministerio de Ciencia y Tecnología/FEDER, TIC2003-02737-C02-02.

<pre><%= home.getHTMLTableHeader() %> <% Collection col=home.findAll(); Iterator it=col.iterator(); while (it.hasNext()) { Person ejb=(Person) it.next(); out.print(ejb.toHTMLTableRowString()); } %></pre> <p style="text-align: center;">With EJBs</p>	<pre><%= FPPerson.getHTMLTableHeader() %> <% FPPerson.getList(bd, out, ""); %></pre> <p style="text-align: center;">With Standard Java classes</p>
---	---

Tabla 1. Secuencias de código Java embebidos en páginas JSP

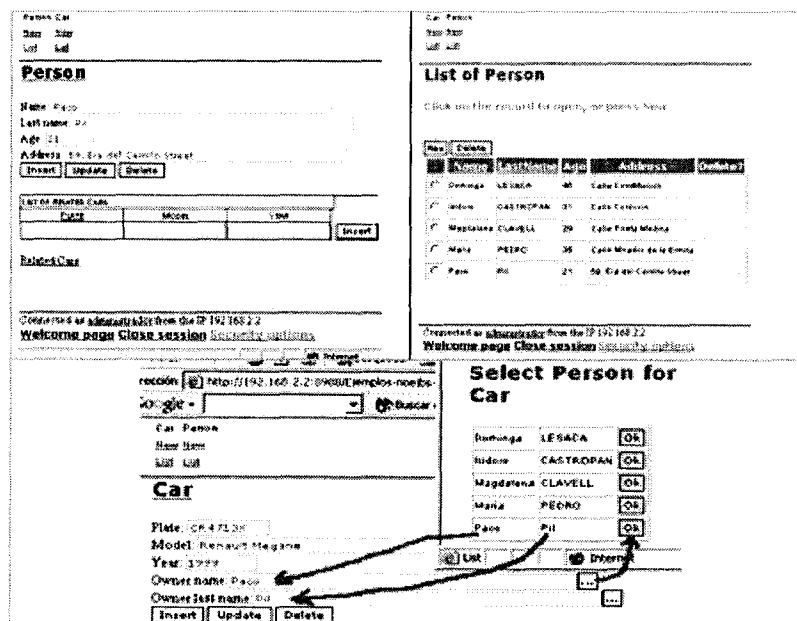


Figura 11. Algunas páginas generadas para la capa de presentación del negocio

