Generación Automática de Código a partir del modelo de datos

APU Walsamakis, Máximo[1]*, APU Mansutti, Marcos*[2]*,*
Lic. Rodolfo Bertone[3]*, Lic. Raul Champredonde*[4]

III LIDI – (Instituto de Investigación en Informática) [5]
Facultad de Informática – UNLP
UNPSJB[6]

Resumen

Una de las causas más comunes de insatisfacción del usuario/cliente respecto del desarrollo de Sistemas de Información (SI) está relacionada al tiempo de respuesta que se obtiene por parte de los desarrolladores. Por otro lado, los desarrolladores se encuentran ante la disyuntiva que algunas de las herramientas que se utilizan generalmente en el mercado no poseen características adecuadas para derivar, a partir de la especificación del modelo de datos, el código básico del sistema.

En nuestro caso, el desarrollo de aplicaciones bajo Delphi nos fuerza a desarrollar toda la interfaz de usuario para la administración de las actualizaciones clásica sobre la BD. Esta tarea, además de ser monótona, insume tiempo de desarrollo que podría utilizarse en otras etapas de la Ingeniería de Software o en el desarrollo de las reglas de negocio específicas de la aplicación.

Los objetivos perseguidos por el trabajo que se presenta a continuación consisten en complementar Delphi con la posibilidad de generación automática del código básico de la aplicación a partir del modelo de datos definido para un SI.

Como se describirá luego, para lograr independizarse del DBMS en el que se implemente el modelo de datos, se utiliza una definición del modelo en XML.

Palabras Clave

Derivación automática de Código. Modelado de datos. Bases de Datos. XML. Ingeniería de Software.

[1] UNPSJB - {talenor@uolsinectis.com.ar}
[2] UNPSJB
[3] Profesor Adjunto Dedicación Exclusiva – Facultad de Informática – UNLP {pbertone@lidi.info.unlp.edu.ar}
[4] Profesor Adjunto Dedicación Exclusiva - Facultad de Informática – UNLP {rchampre@lidi.info.unlp.edu.ar}

[5]Calle 50 y 115 – Primer Piso – La Plata (1900) – Buenos Aires – Argentina TE +54 221 4227707
[6] Universidad Nacional de la Patagonia San Juan Bosco - Sede Comodoro Rivadavia - Rutas 3 y Provincial 1.

Introducción

Los requerimientos de los usuarios y/o clientes de SI están en continua evolución. Además de esta continuidad, los clientes consideran, normalmente, que los tiempos empleados para el desarrollo integral del SI resultan generalmente excesivos y no se condicen con sus necesidades. [Pfleeger 2002]

Para mantener dentro de la planificación el desarrollo de un SI se puede minimizar, entre otros, el tiempo necesario para realizar la codificación. Si bien este tiempo de codificación es mínimo dentro del tiempo de vida de un sistema, dentro de la etapa de codificación, las tareas repetitivas no específicas del dominio de aplicación ocupan generalmente entre un 50 y un 60% del tiempo. Además, la puesta a punto y depuración de la funcionalidad y la interfaz de usuario resulta en valores temporales que no pueden considerarse despreciables. [Soomerville 2002]

Por otro lado, un aspecto importante de toda aplicación, en particular cuando se trata de un SI, es la coherencia y ortogonalidad de la interfaz con el usuario, debiendo esta presentar la información e interactuar con el usuario en forma homogénea y consistente para permitir la concentración en el manejo de la información y no en el uso del sistema.

Este es un objetivo valioso que normalmente resulta tedioso para los desarrolladores. Es aquí donde resultan particularmente útiles los lenguajes de denominados de cuarta generación (4GL) (como por ejemplo Clarion [Clarion 4]) y las herramientas CASE de generación de código.

Un generador de código automático es una herramienta que deriva, a partir de determinados patrones, el código fuente de una aplicación. El uso de estas herramientas reduce el tiempo que se necesita para el desarrollo del software como así también asegura que el grado de errores de programación permanezcan acotados, reduciendo por tanto los tiempos de depuración y puesta a punto. Los 4GL constan de procedimientos que generan el código fuente en función de lo expresado en el diseño de la aplicación o modelo de datos. Para esto, el usuario especifica la funcionalidad del programa, o parte del mismo, y la herramienta determina cómo realizar dicha tarea. [Pressman 2001]

Estas herramientas (Wizards de lenguajes procedurales, herramientas CASE, lenguajes 4GL) son útiles en ámbitos donde se necesita tanto la generación de código puro (sentencias SQL, PHP, o de otros lenguajes), como también la creación de interfaces de usuario (pantallas, reportes, formularios) o ambas.

Dentro del ámbito de la generación automática de código se plantean diferentes acercamientos sobre el tema. Existen propuestas relativas a la implementación de soluciones a partir de diagramas UML [Larman 2003]. Con esta aproximación, se intenta traducir los esquemas de relaciones entre clases que representan a una base de datos a clases e interfaces implementadas en un lenguaje en particular. Se pretende tener en cuenta temas tales como la generación de aplicaciones cliente-servidor, la modularización del código generado, la separación de las interfaces de usuario y de las clases, y la utilización de estándares de codificación.

Además, existen acercamientos relacionados con la generación de interfaces de usuario, donde se pretende seguir un esquema que permita unificar los criterios de representación de los datos. Estos esquemas asocian distintas interfaces de usuario sobre los datos de forma que, para realizar una misma operación sobre los datos, siempre se utilice la misma forma de representación. [Dix 2002]

Al tratar de aplicar el acercamiento mencionado anteriormente sobre una Base de Datos, se necesitan mecanismos que generen la representación del contenido de cada una de las tablas que esta contiene. Además, cada una de estas interfaces visuales deben interactuar tanto entre ellas como con el usuario. Esta interacción se debe ajustar a las relaciones de cardinalidad entre las tablas como así también de la representación individual de cada campo. Toda tabla contiene un conjunto de registros, donde cada uno de estos se puede modificar, eliminar o insertar. Por

otra parte se debe permitir la visualización general del contenido de la tabla, donde se puedan apreciar la mayor cantidad de registros simultáneamente.

Las herramientas que plantea el esquema anterior permiten al programador interactuar con aspectos generales del lenguaje en el que están implementadas. Estas permiten la utilización de distintos controles para la manipulación de los datos para lograr la adaptación del proceso final de la herramienta a las preferencias del usuario.

Modelado de Datos

Un modelo de datos es la representación de la estructura que tendrá la información en un problema particular [Date 2001]. En el caso de los sistemas de gestión el modelo de datos, generalmente representado a través de un modelo Entidad Relación, se comienza a armar en el momento que se realiza el análisis del problema. De esta forma se obtiene el denominado modelo ER conceptual, el cual representa la naturaleza de la información del problema sin entrar en detalles de implementación. A partir de aquí, y a medida que se evoluciona en el desarrollo del sistema, se deriva un modelo ER lógico, donde se toman decisiones de diseño acordes con los requerimientos detallados del problema [Batini et al., 1989]

Como una de las últimas etapas del diseño del SI, se genera el modelo relacional que representa la BD del problema. A partir de esta etapa comienza la codificación del SI. Una vez alcanzado este punto, es interesante poder derivar todo el código de actualización sobre la BD como una operación que no insuma tiempo de desarrollo al programador. [Hoffer 2002]

XML

XML (eXtensible Markup Language) en un lenguaje de etiquetas extensible con un formato basado en el etiquetado textual para documentos y datos. XML no es realmente un nuevo lenguaje, es un metalenguaje usado para definir a otros lenguajes, actuando como protocolo integrador entre aplicaciones. Permite crear documentos bien estructurados y como resultado todo lenguaje basado en XML también será bien estructurado, lo que significa que los datos en XML son más fáciles de utilizar, así XML es una nueva manera de comunicarse a través de Internet o entre programas, porque permite a los negocios y sus sistemas de computadoras comunicarse más fácilmente. [Kleiner 2002] [Ray 2001]

Propuesta de desarrollo

El trabajo realizado tiene como objetivo, entonces, ampliar el marco de trabajo de Delphi, incorporando al mismo la capacidad de derivación automática de código. Para ello, como se indicó, se utilizó como base de información el modelo de datos del problema definido expresado en XML. La motivación de este trabajo es reducir el tiempo de desarrollo de aplicaciones cuando se utiliza esta herramienta y un motor de BD como puede ser SQL Server o My-SQL.

Las motivaciones particulares para el desarrollo de esta herramienta, en nuestro caso, se debieron a que en nuestro entorno de trabajo estamos desarrollando aplicaciones de transferencia a terceros las cuales estaban siendo demoradas (entre otros factores) por el tiempo insumido en generar las actualizaciones generales de la BD. Además, este tipo de tareas se encuentra presente en la mayoría de los sistemas comerciales, de aquí se deriva la necesidad primera del desarrollo.

La elección de XML para expresar el modelo de datos se debe particularmente a las características expuestas anteriormente. XML es un meta-lenguaje que permite crear documentos muy bien estructurados. Esta característica lo hace muy útil cuando se intenta definir todas las características del modelo de datos.

Como se explicará más en detalle en el apartado siguiente el esquema de trabajo planteado se ajusta al desarrollo de estas tres componentes para la herramienta:

- Browsers: interfaz con listas de datos que permiten ser filtrados, buscados, seleccionados, etc. Cada Browser representa a una tabla o una vista de la BD en particular.
- Formularios de Actualización: interfaz donde se presenta un registro particular de una tabla. Sobre ella se puede realizar el ABM clásico de los datos contenidos. Admite la utilización de controles o componentes donde la representación de los datos es muy particular (por ejemplo, la utilización de objetos que permitan ver "calendarios" para la representación de fechas; la utilización de componentes que permitan la edición de textos, permitiendo aplicarles formatos o cambiar los tipos de letras; etc.).
- Reportes: interfaz que permite generar los listados de presentación de los datos contenidos en las tablas de la BD

Junto con estas interfaces visuales se perfiló un esquema para la toma de información por parte la herramienta. Para que esta herramienta cumpla su función, la cantidad de parámetros que se dejen a especificación por parte del programador debía ser mínima. Si el tiempo que el programador utiliza sobre la herramienta para tomar decisiones es largo, la utilidad de la misma seria cuestionada.

Luego de indicar a esta los parámetros necesarios, su ejecución debe completarse con la creación de una aplicación completa, donde no existan errores de sintaxis o diferencias entre interfaces del mismo tipo.

Trabajo realizado

En esta sección se presenta, en forma somera, el trabajo realizado. El mismo, básicamente, fue divido en dos etapas: la primera consistió en la representación del modelo de datos en un lenguaje intermedio que pudiera ser reconocido desde Delphi y a partir del cual se pudiera generar la aplicación. La segunda etapa consistió en el desarrollo de la herramienta (wizard) que permite generar el código de acuerdo a las expectativas del programador. Para mayores detalles respecto de las decisiones tomadas para generar el modelo y como se derivó la herramienta se puede consultar a [Walsamakis et al., 2004].

Modelo de datos

El Modelo de Datos es la representación de la estructura de la base de datos. Este contiene información sobre la descripción estructural de las tablas que habitan en ella como así de las relaciones entre estas. Cada uno de los campos que componen a las tablas son identificados mediante una estructura para poder especificar datos relevantes sobre estos tales como: nombre que lo identifica, tipo de datos que almacena, restricciones de ingreso, máscara para la representación de información, tamaño según el tipo, etc.

La representación en XML de las estructuras de la base de datos independiza al Wizard de las representaciones propias de cada DBMS que se utilice

Para crear un DTD que represente el modelo de Datos es necesario analizar la estructura de una Base de Datos en general. Se deberán identificar aquellos elementos importantes que conformarán al DTD. Es necesario determinar aquel elemento principal ("elemento raíz") que contendrá al resto del Documento XML. Se ha definido como elemento raíz a "ModeloDeDatos".

Al analizar la estructura de la base de datos, se logra extraer de su estructura dos componentes primordiales. Estos son las tablas, fundamentales para la generación de las interfaces de usuario, como así también a las relaciones que hay entre ellas, las cuales indican como se deben comportar dichas interfaces mencionadas. A raíz de esto de crean dos nuevos elementos "Tabla" y "Relación".

Junto con la declaración del Elemento raíz se define la cantidad de ocurrencias para cada elemento de la siguiente forma:

- Se define que el elemento “Tabla” debe existir dentro del elemento “ModeloDeDatos” por lo menos una vez, por lo que se utiliza el carácter “+”.
- Se define que el elemento “Relación” no tiene restricciones de ocurrencia, por lo que se utiliza el carácter “*” que implica cualquier cantidad, incluyendo cero.

Una vez descritos los bloques principales que componen al Modelo de Datos se procede a analizar la estructura de cada uno de ellos en profundidad. No se especifica detalladamente este análisis, pero se debe notar que asociado con cada tabla deben estar las claves definidas (principal, candidatas y/o secundarias), los campos que componen la tabla (junto con el dominio, valor por defecto definido, si se acepta valor nulo, y las demás características que se pueden definir sobre un atributo).

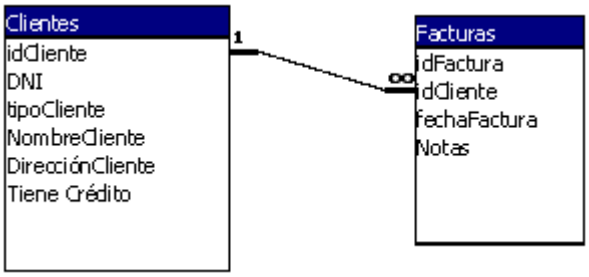
Respecto de la definición de la/s clave/s (denominada índice, genéricamente) las mismas deben contener todas las características propias que la definen, estas son: campos y tipo de ordenamiento (ascendente o descendente).

Además, la segunda componente en importancia definida anteriormente lo representan las relaciones existentes entre las tablas. Este dato no es menor dentro del DTD, las interfaces que se generarán posteriormente mediante la herramienta deben reflejar las relaciones del modelo. Para definir una relación se necesitan dos tablas: origen y destino, las cuales están “relacionadas” por uno o más campos, los cuales deben estar definidos en el DTD.

Por último, se debe especificar el tipo de relación definida: uno a uno (donde cada registro de la tabla origen tiene un solo registro asociado en la tabla destino) o uno a muchos (donde cada registro de la tabla origen puede tener uno o más registros asociados en la tabla destino). No se contemplan, por razones obvias, las relaciones muchos a muchos, ya que el modelo de datos relacional las ha eliminado en etapas anteriores del diseño.

En resumen, el esquema siguiente representa el formato general de un DTD y luego como quedan dos tablas definidas mediante dicho formato:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!ELEMENT ModeloDeDatos (Tabla+,Relación*)>
<!ELEMENT Tabla (NombreTabla,Campo+,Indice+)>
<!ELEMENT Campo (Nombre,LongitudDisplay?,MensajeValidación?,MáscaraDeEntrada?,ListaDeValores?)>
<!ELEMENT NombreTabla (#PCDATA)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT LongitudDisplay (#PCDATA)>
<!ELEMENT MensajeValidación (#PCDATA)>
<!ELEMENT MáscaraDeEntrada (#PCDATA)>
<!ELEMENT ListaDeValores (Valor)+>
<!ELEMENT Valor (#PCDATA)>
<!ELEMENT Indice (CampoIndice+)>
<!ELEMENT CampoIndice (#PCDATA)>
<!ELEMENT Relación (TablaOrigen,TablaDestino,ClavesTablaOrigen,ClavesTablaDestino)>
<!ELEMENT TablaOrigen (#PCDATA)>
<!ELEMENT TablaDestino (#PCDATA)>
<!ELEMENT ClavesTablaOrigen (CampoClave)+>
<!ELEMENT ClavesTablaDestino (CampoClave)+>
<!ELEMENT CampoClave (#PCDATA)>
<!ATTLIST
        Relación tipo (unoAuno | unoAmuchos) "unoAmuchos" >
<!ATTLIST Indice
        nombre CDATA #REQUIRED
        tipo (primario | secundario | foraneo) #REQUIRED >
<!ATTLIST CampoIndice
        ord (asc | des) "asc" >
<!ATTLIST Campo
        tipoDeCampo (cadena | numérico | booleano | fechaHora | memo | blob) "cadena" >
```




```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ModeloDeDatos SYSTEM "ModeloDeDatos.dtd">
<ModeloDeDatos>
 <Tabla>
  <NombreTabla>Facturas</NombreTabla>
  <Campo tipoDeCampo="numérico">
   <Nombre>idFactura</Nombre>
   <LongitudDisplay>8</LongitudDisplay>
  </Campo>
  <Campo tipoDeCampo="numérico">
   <Nombre>idCliente</Nombre>
   <LongitudDisplay>8</LongitudDisplay>
  </Campo>
  <Campo tipoDeCampo="fechaHora">
   <Nombre>fechaFactura</Nombre>
   <LongitudDisplay>10</LongitudDisplay>
  </Campo>
  <Campo tipoDeCampo="memo">
   <Nombre>Notas</Nombre>
   <LongitudDisplay>255</LongitudDisplay>
  </Campo>
  <Indice nombre="PrimaryKey" tipo="primario">
   <CampoIndice ord="asc">idFactura</CampoIndice>
   <CampoIndice ord="asc">idCliente</CampoIndice>
  </Indice>
  <Indice nombre="idCliente" tipo="foraneo">
   <CampoIndice ord="asc">idCliente</CampoIndice>
  </Indice>
  <Indice nombre="idFactura" tipo="secundario">
   <CampoIndice ord="asc">idFactura</CampoIndice>
  </Indice>
 </Tabla>
 <Tabla>
  <NombreTabla>Clientes</NombreTabla>
  <Campo tipoDeCampo="numérico">
   <Nombre>idCliente</Nombre>
   <LongitudDisplay>8</LongitudDisplay>
  </Campo>
  <Campo tipoDeCampo="numérico">
   <Nombre>DNI</Nombre>
   <LongitudDisplay>11</LongitudDisplay>
   <MensajeValidación>El número de documento es  incorrecto</MensajeValidación>
   <MáscaraDeEntrada>##"."###"."###</MáscaraDeEntrada>
  </Campo>
  <Campo tipoDeCampo="cadena">
   <Nombre>tipoCliente</Nombre>
   <LongitudDisplay>14</LongitudDisplay>
   <ListaDeValores>
    <Valor>Sin Problemas</Valor>
    <Valor>Debe</Valor>
    <Valor>Incobrable</Valor>
   </ListaDeValores>
  </Campo>
  <Campo tipoDeCampo="cadena">
   <Nombre>NombreCliente</Nombre>
   <LongitudDisplay>250</LongitudDisplay>
```

```
    </Campo>
    <Campo tipoDeCampo="cadena">
      <Nombre>DirecciónCliente</Nombre>
      <LongitudDisplay>250</LongitudDisplay>
    </Campo>
    <Campo tipoDeCampo="booleano">
      <Nombre>Tiene Crédito</Nombre>
    </Campo>
    <Indice nombre="PrimaryKey" tipo="primario">
      <CampoIndice ord="asc">idCliente</CampoIndice>
    </Indice>
    <Indice nombre="idCliente" tipo="secundario">
      <CampoIndice ord="asc">idCliente</CampoIndice>
    </Indice>
    <Indice nombre="otroIndice" tipo="secundario">
      <CampoIndice ord="asc">DNI</CampoIndice>
      <CampoIndice ord="des">NombreCliente</CampoIndice>
    </Indice>
  </Tabla>
  <Relación tipo="unoAmuchos">
    <TablaOrigen>Clientes</TablaOrigen>
    <TablaDestino>Facturas</TablaDestino>
    <ClavesTablaOrigen>
      <CampoClave>idCliente</CampoClave>
    </ClavesTablaOrigen>
    <ClavesTablaDestino>
      <CampoClave>idCliente</CampoClave>
    </ClavesTablaDestino>
  </Relación>
</ModeloDeDatos>
```

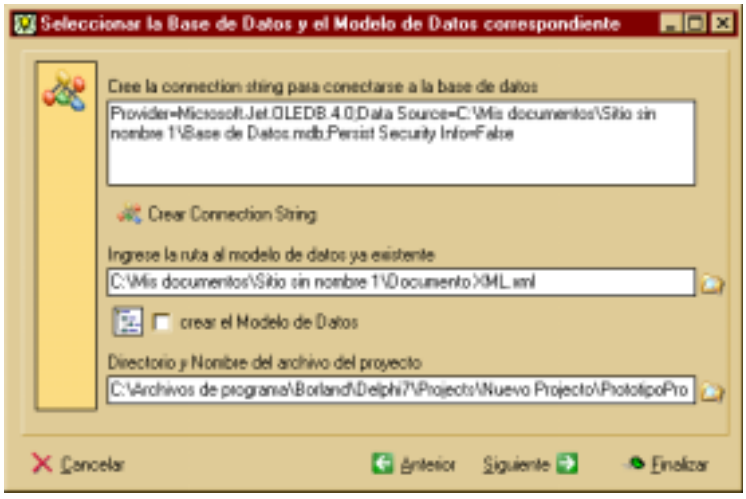
Desarrollo de la herramienta

Para que el herramienta lograse generar las interfaces de usuario, que formarían parte del programa, se debió crear un conjunto de clases que representasen tanto el contenido del Modelo de Datos en XML, como así también la información necesaria para su aplicación dentro del entorno de programación.

Este conjunto de clases permite interpretar el contenido del DTD en XML para poder generar las distintas etapas y partes de la aplicación. Además permite ver tanto el esquema de la base de datos (representada en el archivo XML) como también las relaciones entre las distintas partes que componen a una base de datos (tablas, índices, relaciones, campos, etc.).

El objetivo de poder utilizar distintos DMBSs, requiere de la utilización de alguna tecnología de accesos a base de datos que permitan esta transparencia. Como la aplicaciones objetivo inicialmente son aplicaciones Windows, se eligió un conjunto de controles DBGo que son compatibles con Delphi y encapsulan el comportamiento de las interfaces ADO, las que permiten el acceso a base de datos a través de ODBC y drivers desarrollados por los propios fabricante del DBMS. ADO se encuentra altamente distribuido dentro de los sistemas operativos Windows (componentes MCDAC y Jet Engine).

En la utilización del Wizard desarrollado, inicialmente el programador debe informar algunos datos generales necesarios completando el siguiente formulario:



- Connection string: Se necesita crear un string de conexión que indique a los componentes DBGo como conectarse a la Base de Datos. Puede escribir directamente el contenido de esta, o presionar en el botón “Crear Connection String” para que se active el cuadro de diálogo estándar de Windows para realizar dicha tarea.
- Ruta del Modelo de Datos: se debe indicar la ruta donde se encuentra el archivo XML que representa al modelo de datos de la base de datos. Si este no ha sido generado, se puede seleccionar “crear el modelo de Datos” para que la aplicación invoque a un módulo externo para crearlo.
- Directorio y nombre del Proyecto: Ingrese la ruta del directorio donde se almacenarán todas las unidades generadas y el nombre que tendrá el nuevo proyecto.

La segunda etapa de la herramienta consiste en la selección del formato de los Browsers que se van a generar, aquí es posible obtener una interfaz que presente para cada tabla los datos ordenados de acuerdo a cada índice definidos, o por le contrario permitirá seleccionar aquellos sobre el cual se realizará la presentación.

Una vez cubiertas las dos etapas anteriores, el proceso de generación de la aplicación comienza permitiendo obtener, como resultado final el esquema de actualizaciones sobre talas definidas en el modelo de datos.

Resultados obtenidos

La herramienta desarrollada fue probada con el desarrollo de un sistema para la administración de la información de los matriculados de un Colegio Profesional de la Pcia. De Buenos Aires.

Para hacer una medición del nivel de influencia del uso de esta herramienta con respecto a una metodología de desarrollo tradicional, se caracterizó el sistema por medio de la cantidad de browsers y ABMs necesarios. Cada browser además fue clasificado según su complejidad dependiendo de la cantidad y tipos de relaciones de la tabla subyacente con las demás, en complejidad baja, media y alta.

Se deja expresamente de lado las reglas de negocio específicas de la aplicación.

El sistema requiere de 22 browsers y sus respectivos 22 ABMs, de los cuales:

- 7 son de complejidad baja
- 3 son de complejidad media
- 12 son de complejidad alta

El tiempo aproximado de desarrollo usando una metodología de programación tradicional fue de aproximadamente algo menos de 100 horas/hombre.

En contraste, el tiempo de generación del código equivalente contando previamente con el modelo de datos expresado en XML, el cual se obtiene automáticamente con una herramienta desarrollada a tal efecto, no superó los 5 minutos. A este contraste temporal, se debe considerar además la homogeneidad en la apariencia y comportamiento del resultado obtenido, lo que redunda en una mayor satisfacción de los usuarios [Jones 1994].

Conclusiones

Se ha desarrollado un conjunto de elementos que dotan a Delphi de la posibilidad de generar automáticamente código de actualizaciones sobre el modelo de datos de una forma rápida y uniforme que permite acortar el plazo de desarrollo de un SI. De esta forma el tiempo de desarrollo se reduce prácticamente a aquel que se necesite para el desarrollo del código correspondiente a las reglas de negocio específicas de una aplicación en particular.

Entre los trabajos futuros se contempla seguir desarrollando aplicaciones que permitan obtener el código XML del modelo de datos directamente de la definición del mismo sobre el DBMS. En particular, actualmente se puede derivar dicho código desde un modelo definido sobre SQL-Server y MySQL.

Bibliografía

[Batini et al., 1989] Diseño conceptual de Bases de Datos. Batini, Navathe, Cieri. Addison Wesley. 1989.

[Clarion 4] Manual de Referencia. Top Speed.

[Date 2001] Introducción a los Sistemas de Bases de Datos. C.J. Date. Pearson Education 2001.

[Dix 2002] Human-Computer Interaction (2nd Edition) Alan J. Dix, Janet E. Finlay, Gregory D. Abowd, Russell Beale, Janet E. Finley. 2002

[Hoffer, 2002] Modern Database Management. Sixth Edition. J. Hoffer, M. Prescott, F. McFadden. Prentice Hall 2002.

[Jones 1994] Assessment and Control of Software Risks. Capers Jones. Yourdon Press Computing Series. 1994

[Kleiner et al., 2002] A*utomatic Generation of XML DTDs from Conceptual Database Schema.* Carsten Kleiner, Udo W. Lipeck para Universität Hannover, Institut für Informatik, Lange Laube 22, 30159 Hannover, Germany

[Larman 2003] UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Craig Larman. Prentice Hall. 2003

[Pfleger 2002] Ingeniería de Software. Teoría y Prácitca. Shari Pfleeger. Prentice Hall. 2002

[Pressman 2001] Ingeniería de Software. Un enfoque práctico. Roger Pressman. McGraw Hill Int. 2001.

[Ray 2001] *Learning XML.* Erik T. Ray, First Edition, January 2001, ISBN: 0-59600-046-4, 368 pages

[Sommerville 2002] Ingeniería de Software. 6ta Edición. Ian SOmmerville. Addison Wesley 2002

[Walsamakis et al., 2004] Herramienta para la generación automática de código Delphi a partir de un modelo de datos expresado en XML. M. Walsamakis, M Mansutti. Trabajo de Grado. Licenciatura en Informática. UNPSJB. Directores: Rodolfo Bertone, Raul Champredonde. Comodoro Rivadavia 2004.