View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Centro de Servicios en Gestión de Información

Integrando UML y DSL en el enfoque MDA

Daniel Giulianelli[1], Claudia Pons[2], Rocío Rodríguez[1]
Pablo Vera[1], Víctor Fernandez[1]

[1]Universidad Nacional de La Matanza
Departamento de Ingeniería e Investigaciones Tecnológicas
Florencio Varela 1903, San Justo, Buenos Aires, Argentina

[2] Universidad Nacional de La Plata
Facultad de Informática
LIFIA-Laboratorio de Investigación y Formación en Informática Avanzada
Calle 50 y 150 La Plata, Buenos Aires, Buenos Aires, Argentina

{dgiulian, rrodri, pablovera, vfernandez }@unlam.edu.ar
cpons@lifia.info.unlp.edu.ar

Resumen. En algunos trabajos académicos surge la disyuntiva de utilizar UML (Unified Modeling Language) ó DSL (Domain Specific Lenguage) para modelar un determinado artefacto. UML es un lenguaje de propósito general el cual en un nivel de abstracción elevado resulta de gran aplicabilidad, pero cuando se comienza a bajar dicho nivel de abstracción y se requiere comenzar a modelar características propias de un dominio, UML debe ser adaptado. Es posible adaptar a UML generando un perfil propio para dicho dominio pero esta actividad resulta compleja y en algunos dominios son muy pocos los elementos y diagramas existentes que son directamente aplicables y por lo tanto es necesario realizar una gran cantidad de extensiones para lograr modelar el dominio. En cambio DSL es un lenguaje más simple de aplicar a un dominio específico. En este trabajo se presenta una propuesta que permite dentro del enfoque MDA (Model-Driven Architecture) utilizar UML y DSL en distintos niveles de abstracción y generar mediante transformaciones el código fuente de una determinada aplicación.

Keywords: Modelado, MDA, UML, DSL, WAP

1 Introducción

Actualmente los sistemas son muy disimiles unos de otros, es por ello que al modelar un sistema que pertenezca a un determinado dominio UML [4] resulta ser muy amplio y complejo de adaptarse a las características particulares de dicho dominio. Al momento de modelar el sistema, será necesario analizar el vocabulario de UML (simbología e incluso diagramas que pueden ser aplicados) y extender el lenguaje por ejemplo por medio de estereotipos y crear un profile que permita modelar las características no nomencladas. DSL [5] ha sido creado con la idea de poder modelar características particulares de dominios.

En este artículo se propone modelar un sistema con el enfoque MDA [7], [8] utilizando a UML como lenguaje de modelado que permite analizar al sistema desde un punto de abstracción alto y a DSL en un nivel más bajo de abstracción y más cercano a la codificación específica en una determinada plataforma.

MDA es un enfoque ampliamente aceptado para el desarrollo de sistemas de software complejos. "Es una iniciativa del OMG (Object Management Group), que representa un nuevo paradigma de desarrollo de software donde los modelos guían todo el proceso de desarrollo…" [4]

MDA propone el uso de modelos en todas las fases de desarrollo, desde la especificación y análisis hasta la implementación. La transformación de modelos es la base de MDA; comenzando por un modelo independiente de la plataforma el objetivo es lograr, en cada paso, modelos más específicos. "Los modelos son creados en diferentes niveles de abstracción separando los aspectos del negocio de los detalles técnicos de la solución de software que se va a implementar. Básicamente tres diferentes tipos de modelos son construidos, un modelo que contiene las especificaciones de negocio, un modelo de alto nivel de la plataforma y uno que incluye los detalles técnicos de la plataforma destino" [3]

"MDA fue establecida como una arquitectura para el desarrollo de aplicaciones; tiene como objetivo proporcionar una solución para los cambios de negocio y de tecnología, permitiendo construir aplicaciones independientes de la implementación; representa un nuevo paradigma en donde se utilizan modelos del sistema, a distinto nivel de abstracción, para guiar todo el proceso de desarrollo" [6] . A continuación se definen cada uno de los modelos de MDA tal como se muestra en la figura 1[7]:

- *CIM:* Es un modelo independiente de lo computacional. No muestra detalles de la estructura de un sistema. También suele ser denominado modelo de dominio, y para su especificación se utiliza un vocabulario que es familiar a los practicantes del dominio en cuestión. Se focaliza en el contexto

  El CIM juega un rol importante al unir la brecha entre:
  - aquellos que son expertos en el dominio y sus requerimientos
  - aquellos que son expertos en el diseño y construcción de artefactos

  En este nivel se representa exactamente qué es lo que se espera que el sistema haga, pero oculta la información de la tecnología o como será implementada.
- *PIM:* Es un modelo independiente de la plataforma. Esto se puede lograr a través de un modelado que no esté enfocado a una determinada plataforma sino que realice una implementación abstracta de los detalles técnicos necesarios para su construcción.
- *PSM:* Es un modelo específico de la plataforma. Combina las especificaciones del PIM con los detalles que indican como ese sistema utiliza un tipo particular de plataforma.

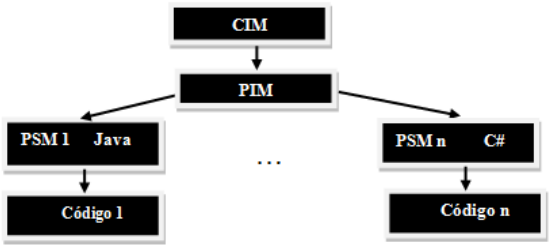


Fig. 1. Esquema MDA

2 Integración de UML y DSL en el enfoque MDA

Este trabajo se enfoca en utilizar dos lenguajes de modelado (UML-DSL) aprovechando las ventajas de ambos aplicándolos a distintas etapas del modelado.

- UML - Lenguaje de propósito general
- DSL - Lenguaje de propósito específico

Los lenguajes de dominio específico, como por ejemplo DSL, son una alternativa a UML para modelar aplicaciones. A diferencia de UML no tienen estructuras generales sino que para modelar cada tipo de aplicación se debe definir un DSL específico con las entidades que se necesiten modelar. Esto hace que el lenguaje sea más acotado y específico.

UML al ser de propósito general se vale de estereotipos y profiles para poder adaptarse lo más posible a dominios específicos mientras que un DSL nace específicamente para dicho dominio. Los DSL al ser más acotados son más propicios para la generación de código.

Es posible utilizar UML para modelar la aplicación de forma genérica (CIM) e independiente de la plataforma (PIM) y tener un DSL de más bajo nivel ya dependiente de la plataforma (PSM) que permita de forma más sencilla la generación de código. Por lo tanto se podrían tener varios PSM modelados en DSL para cada plataforma sobre la cual se desee generar código (ver figura 2).

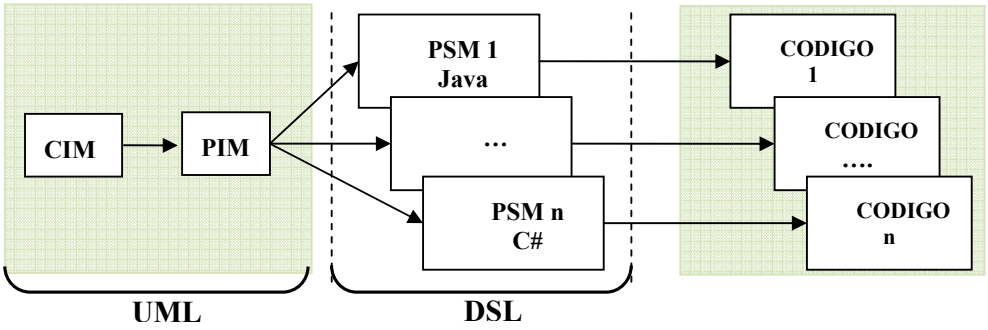


Fig.2. Aplicando UML y DSL al enfoque MDA

"DSL eleva el nivel de abstracción más allá de los lenguajes de programación actuales a través de la especificación de la solución, utilizando directamente conceptos de dominio del problema. El código fuente es generado desde este nivel de especificación. Esta automatización es posible porque, ambos el lenguaje y generadores se ajustan los requerimientos de una sola compañía y un dominio"[2]

Para ello se persiguen los siguientes objetivos:

1. Modelar una aplicación bajo el esquema de MDA (Model Driven Architecture);
2. Utilizar UML para modelar el CIM y el PIM;
3. Utilizar DSL para modelar el PSM
4. Desarrollar una herramienta que permita generar automáticamente código a partir de cada PSM construido.

La metodología está compuesta por cuatro etapas:

1. **Generar el CIM:** Para este modelo se ha elegido el diagrama de casos de usos de UML ya que permite ver un bosquejo general de la aplicación y los principales requerimientos del sistema.
2. **Generar el PIM:** Se realizará por cada uno de los casos de usos del CIM un diagrama de actividades en el cual se mostrará la funcionalidad interna. Solo se tomarán aquellos casos de uso que realice en forma directa el usuario ya que para los que dependen de otros, su funcionalidad va a estar incluida en diagrama de actividades correspondiente.
   Estos dos primeros pasos se realizan mediante diagramas propios de UML tal como se indicaba en la figura 2.
3. **Generar el PSM:** Para cada uno de los diagramas de actividades se realiza un DSL enfocado a la plataforma en la cual se quiere desarrollar la aplicación.
4. **Generar el Código Fuente:** Cada una de las construcciones de DSL genera una porción de código especifico a la plataforma elegida, de esta forma al estar las construcciones relacionadas entre sí, generarán un código fuente bastante rico, el cual reducirá el trabajo de programación para obtener el producto final.

3 Modelado de una aplicación

3.1 Plataforma de desarrollo

Como plataforma de la aplicación se ha optado por generar páginas web enfocadas a teléfonos celulares. Estas páginas se basan en un lenguaje denominado WML (wireless markup language) que está diseñado especialmente para dispositivos pequeños, con memoria y capacidades limitadas de procesamiento. Este lenguaje fue diseñado con el objetivo de ser liviano para transferir poca información por la red de datos reduciendo tiempos de espera y costos.

Similar a HTML, WML dispone de una serie de tags que son interpretados por el browser del cliente para mostrar la información.

A continuación se detallan algunos de los tags de WML, especialmente aquellos que luego serán incorporados al modelo DSL que se construye en el ítem 3.2.

- Páginas: son los archivos físicos .wml dentro de los cuales tienen un header que identifica al tipo de documento WML para que sea correctamente interpretado por los browser. El contenido de las páginas debe ser colocado dentro de los tags <wml> </wml>
- Cards (tarjetas): representan el conjunto de datos que serán mostrados en la pantalla del teléfono al mismo tiempo. Una misma página puede contener más de una tarjeta y navegar entre ellas sin necesidad de enviar información por la red, simplemente cambiando la vista en el browser.
- Listas de Selección: son listas de opciones que puede seleccionar el usuario. Al momento de seleccionar una opción se dispara un evento asociado y una variable guarda el valor de la opción seleccionada. Para definir una lista de selección se usa el tag <select> y para cada una de las opciones el tag <option>.

- Controles de ingreso de datos: WML soporta el ingreso de caracteres alfanuméricos mediante el tag <input>, el contenido de este control es ingresado por el usuario y es almacenado en una variable relacionada con el control cuyo nombre se define con el atributo name.
- Acciones: las acciones se representan entre los tags <do></do> Los atributos más importantes que contiene la etiqueta <do> son: type="",label="" y name="".
  1. type="". indica sobre qué botón del navegador se aplica la acción, las más comunes son "accept","prev" y "help".
  2. label="". Texto que aparece asociado a la acción.
  3. name="". Nombre de la acción, es imprescindible si vamos a asignar más de una acción a un mismo tipo.

El contenido de la etiqueta <do> es la tarea que se realizará al seleccionar esa acción, y estas tareas pueden ser:

1. <go href=""/> Permite ir a la dirección indicada en el atributo href.
2. <prev/> Permite ir a la tarjeta anterior, en la historia del navegador.
3. <noop/> Es una acción que no realiza nada.
4. <refresh>...</refresh> Refresca el contenido de la tarjeta actual, volviéndola a pedir al servidor.

Para mayor información sobre WAP se recomienda consultar [12] y [10]

3.2 Modelado

A continuación se describe el problema a modelar. Se trata de un sistema WAP destinado a reparto domiciliario (por ejemplo una empresa de agua mineral que distribuye además otros artículos). Cada empleado tiene la ruta que realizará en un determinado día precargada en su dispositivo móvil. Para simplificar el ejemplo se modelarán las siguientes funcionalidades: Visualización de la ruta: El sistema permite visualizar la ruta en la cual están señaladas las distintas paradas necesarias para visitar a cada cliente; Toma de pedido: En cada domicilio un cliente realiza su pedido, para lo cual es necesario consultar la disponibilidad de dichos productos en el camión.

1. Generar el CIM: En la figura 3 se muestra el modelado por casos de uso.

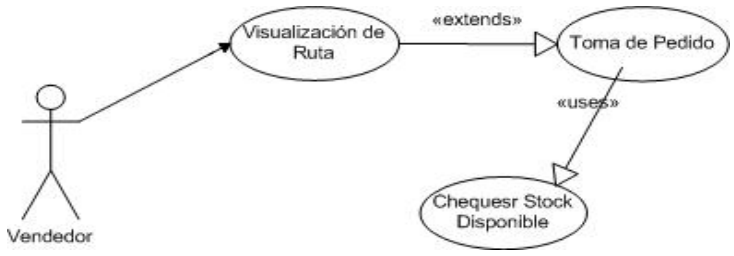


Fig. 3. Modelado UML – Casos de Uso

2. Generar el PIM: Para cada caso de uso que posea el sistema se realiza un diagrama de actividades. En el caso de esta aplicación hay un caso de uso principal "Visualización de ruta" del cual se extiende "Toma de Pedido" caso que utiliza "Chequear Stock disponible". Por ello se realiza un solo diagrama de actividades planteado para el caso de uso principal lo que desencadenará en el

mismo diagrama la necesidad de incluir el modelado de las funcionalidades de los otros dos casos de uso (ver figura 4).

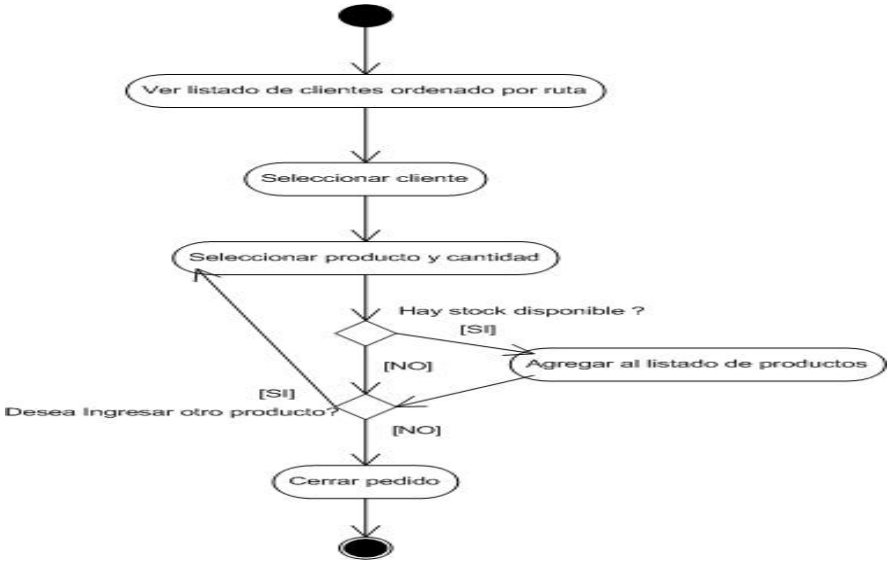
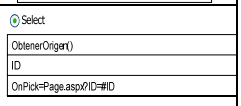
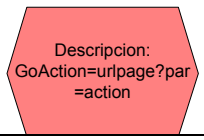


Fig.4. Diagrama de Actividades de un caso de uso

3. Generar el PSM: Para el PSM se debe definir el DSL específico para la plataforma en la que queremos implementar la aplicación. El DSL construido (ver figura 5) contiene los objetos presentados en la tabla 1 que están directamente relacionados con los componentes de WML explicados en el ítem 3.1.

Tabla 1. Objetos de DSL

Página	Representa un archivo físico que representa a la pagina wml	
Tarjeta	Representa los cards de wml	
ListaSelección	Permite definir una lista de opciones para que el usuario pueda elegir una de ellas. Este objeto tendrá, aparte del título, tres atributos: • La función que recupera los datos de la lista • El nombre de la variable interna donde almacenará la opción seleccionada • El evento que se disparará al seleccionar una opción.	
Input	Permite definir un ingreso de datos alfanumérico. Para ello se informa el texto mostrado al usuario y el nombre de la variable interna que contendrá el dato ingresado.	
Acción	Se relacionan con el tag DO de WML, los cuales representan la interacción del usuario. Para las acciones se especifica el texto a mostrar y una URL que permite generar la solicitud sobre el servidor WEB ejemplo: www.mipage.aspx?par=confirm.	

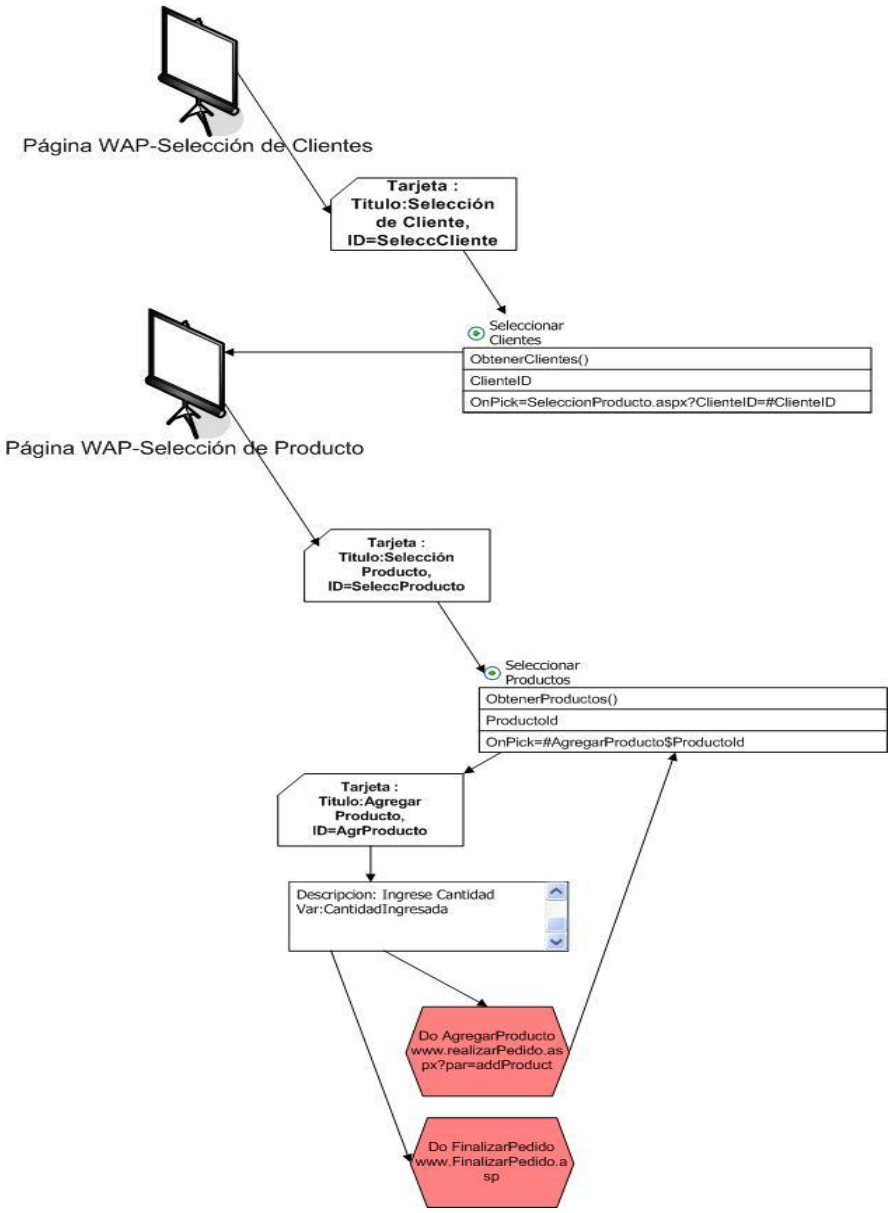


Fig.5. Modelo DSL

4. Generar el código fuente: En base a los elementos que posee el modelado de DSL mostrado en la figura 4 se puede generar automáticamente código para cada uno de ellos, en este ejemplo el modelo cuenta con 5 elementos distintos presentados previamente en la tabla 1. En la figura 6 se muestra una porción de código generada automáticamente para cada uno de estos elementos.

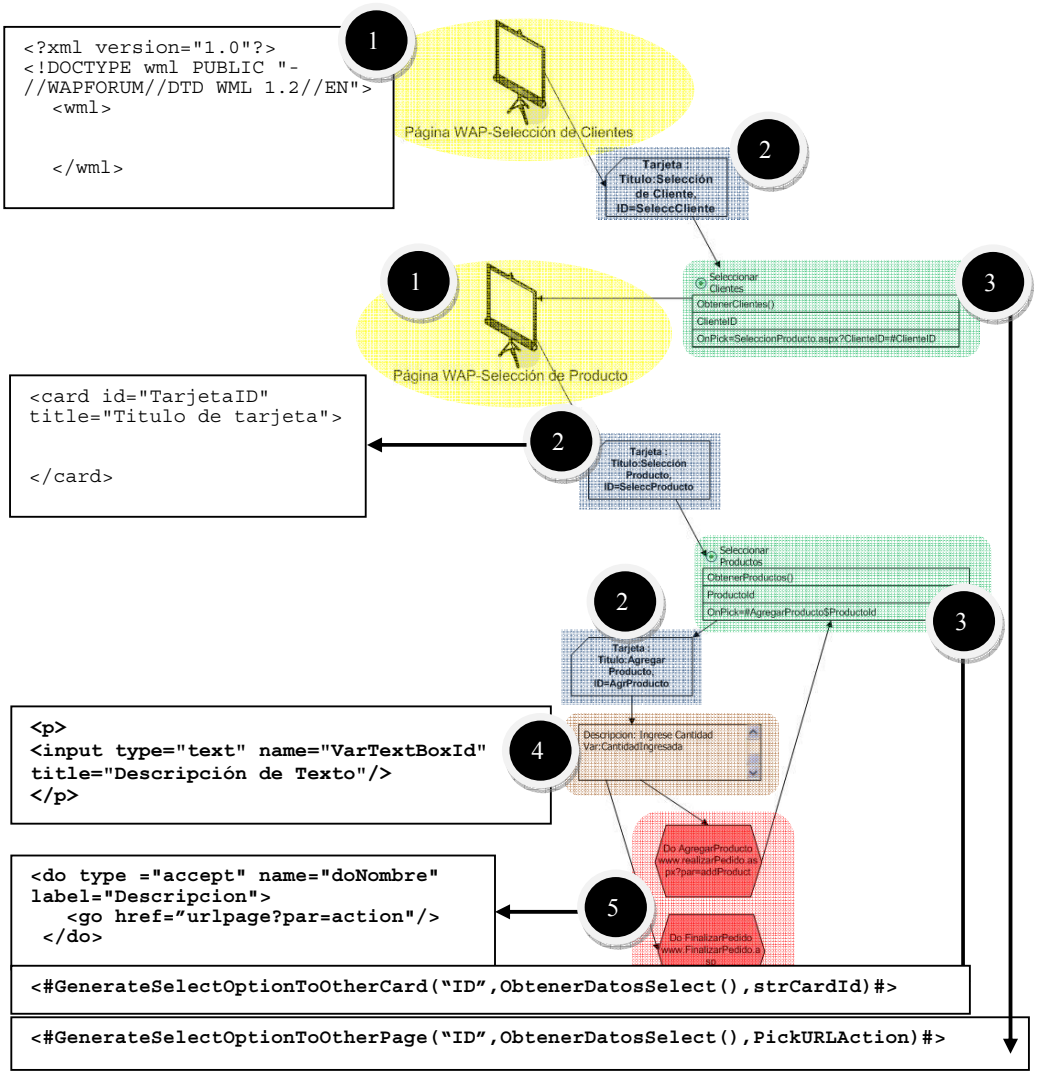


Fig.6. Generación automática de código

Automáticamente se crea en el servidor una función que permite generar dinámicamente las listas de selección. Se muestra a continuación el código generado:

```
Public Function GenerateSelectOptionToOtherPage(ByVal Id  As
string,byref tblSource as DataTable, byval URLPick as string) As
String
        Dim strSelect As String = "<select>"
        For Each dtRow As DataRow In tblWapFormatedTable.Rows
            strSelect+= "<option value=’” + dtRow[Id] + ”’
onPick=’” + urlPick + “?” + Id + “=” + dtRow[Id] + ”’>" +
dtRow("Description") + + "</option>"
        Next
        strSelect += "</select>"
        Return strSelect
 End Function
```

5 Conclusiones

En el presente artículo se plantea que UML y DSL no son excluyentes sino que pueden complementarse y aplicarse al enfoque de MDA, utilizándose UML en un nivel mayor de abstracción y DSL en un nivel más cercano a la implementación.

OMG propone el enfoque MDA y a su vez también el lenguaje UML. Microsoft lanzó DSL como respuesta al modelado de dominios específicos. Quienes utilizaban el enfoque MDA por añadidura utilizaban UML. Sin embargo cuando se intenta adaptar UML a dominios específicos en algunos casos sólo se toma como base una pequeña parte de UML, resultando necesario agregar muchos elementos nuevos con semántica propia haciendo que UML pierda su gran ventaja que es la estandarización. Para reducir esfuerzos de adaptación resulta conveniente usar DSL, ya que nace explícitamente para ser aplicado a un domino en particular pero con la desventaja de no tener un estándar en símbolos y diagramas de modelado. Por lo tanto en los modelos de alto nivel cuando se quiere tener un diagrama que sirva para comunicar las funcionalidades del sistema mediante un estándar que pueda ser fácilmente entendido por las distintas partes resulta conveniente usar UML pero al llevar el modelo a bajo nivel conviene utilizar un lenguaje adaptable al dominio en particular del cual fácilmente pueden aplicarse transformaciones para llegar al código fuente en forma automática derivándolo de los modelos.

Era inevitable comenzar a considerar las ventajas de ambos lenguajes y que estos pudieran complementarse y no verse como alternativas independientes. En algunos papers (por ejemplo [9] [11]) se presenta una aplicación de DSL en el modelo MDA para evitar extender UML a un dominio particular. En este trabajo se presenta el modelado de una aplicación que se toma a modo de ejemplo simplemente a fin de mostrar que ambos lenguajes pueden coexistir y ser aplicados al enfoque MDA.

Referencias

1. Fowler M. Language Workbenches and Model Driven Architecture (2005)

http://martinfowler.com/articles/mdaLanguageWorkbench.html

2. Kelly Steven, Tolvannen Juha-Pekka. Domain-Specific Modeling: Enabling Full Code Generation. ISBN: 978-0-470-03666-2. John Wiley & Sons (2008)

3. Koos de Goede, John Irizarry, "Understanding tool requirements for Model Driven Architecture" (2008)
http://www.omg.org/mda/mda_files/Understanding_MDA_Tool_Requirements2.pdf

4. Lopez E, Gonzalez G, Lopez M, Iduañate R, Proceso de Desarrollo de Software Mediante Herramientas MDA, 2007
http://www.iiisci.org/Journal/CV$/risci/pdfs/C476AI.pdf

5. Microsoft - MSDN, About Domain-Specific Languages, 2008
http://msdn.microsoft.com/en-us/library/bb126278(v=VS.90).aspx

6. Neil Carlos, Pons Claudia, Aplicando MDA al Diseño de un Datawerhouse Temporal, 2006.
http://caeti.uai.edu.ar/archivos/240_APLICANDO_MDA_AL_DISENIO_DE_UN_DATA WAREHOUSE_TEMPORAL.PDF

7. OMG, MDA Guide Version 1.0.1, 2003
http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

8. OMG, Model Driven Architecture, 2009
http://www.omg.org/mda/

9. OMG, Unified Modeling Language, Infrastructure, Version 2.2 (2009).
http://www.omg.org/spec/UML/2.2/Infrastructure/PDF/

10. Tutorial de WAP / WML
http://www.webestilo.com/wml/

11. Vargas Ruiz F., Roda Garcia J., Estevez García,y otros. Generación de Editores Gráficos de Modelos para una Herramienta MDA, Barcelona, (2006)
http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-227/paper15.pdf

12. Wireless Application Protocol Forum. WAP WML Version 1.1. (1999)
http://www.wapforum.org/what/technical/SPEC-WML-19990616.pdf