Un estudio comparativo de dos herramientas MDA: OptimalJ y ArcStyler

J. García Molina, J. Rodríguez, M. Menárguez, M.J. Ortín, J. Sánchez

Departamento de Informática y Sistemas, Universidad de Murcia
Campus de Espinardo 30071 Murcia
jmolina@um.es

Resumen. El desarrollo de software dirigido por modelos es un área que está en su infancia, pero ya han aparecido un buen número de herramientas con la etiqueta MDA. En esta ponencia presentamos los resultados de un trabajo que ha consistido en realizar una comparación de las dos herramientas MDA más extendidas: *OptimalJ* y *ArcStyler*. Primero evaluamos cada una de ellas mediante un análisis de propiedades extraídas de la especificación de MDA, desarrollando la aplicación *Pet Store* con cada herramienta. Posteriormente realizamos el estudio comparativo a partir de las mismas propiedades. La comparación ha puesto de manifiesto importantes decisiones de diseño para una herramienta MDA, como son la forma de anotar el PIM, que el PSM sea implícito o explícito, su naturaleza cerrada o abierta para soportar cualquier tecnología y los mecanismos para la edición y ajuste de las transformaciones.

1 Introducción

En noviembre de 2000, OMG estableció el *framework* MDA (*Model Driven Architecture*) [5,6] como un nuevo paradigma de creación de software en el que los modelos guían todo el proceso de desarrollo. La idea principal que subyace en MDA es "separar la especificación de la funcionalidad de un sistema software de los detalles sobre cómo se lleva a cabo en una determinada plataforma, de manera que los desarrolladores escriban modelos centrados en la lógica de la aplicación y de forma automática se genere el código con los detalles propios de una plataforma". A este nuevo paradigma se le ha denominado *Ingeniería de Modelos* o *Desarrollo Dirigido por Modelos*.

MDA se fundamenta en la "arquitectura de cuatro capas" del OMG [7], en la que MOF es el lenguaje de metamodelado a partir del que se pueden definir lenguajes de modelado como UML. El proceso MDA se basa en transformar modelos independientes de detalles de implementación (modelo PIM) en otros que aportan los aspectos específicos de una plataforma concreta (modelo PSM), hasta llegar al modelo final, esto es el código fuente. MOF permite definir formalmente los lenguajes de modelado y las transformaciones entre modelos, de manera que se puedan construir "compiladores de modelos". Al igual que sucedió con los lenguajes de programación, el desarrollo dirigido por modelos supone un nuevo paso para aumentar el nivel de abstracción y la automatización. También es posible reconocer en el nuevo paradigma

una evolución de los generadores de aplicaciones que tan importante papel han jugado en las dos últimas décadas.

Aunque está claro que el desarrollo de software dirigido por modelos es un área que está en su infancia, ya ha aparecido un buen número de herramientas con la etiqueta "MDA". En este trabajo presentamos los resultados de un estudio comparativo de las dos herramientas MDA más extendidas: *OptimalJ* y *ArcStyler* [1]. Primero se ha realizado una evaluación individual de cada herramienta mediante un análisis de características análogo al empleado en [2] para evaluar OptimalJ. Después se ha contrastado los resultados de cada herramienta. El caso práctico elegido para probar las capacidades de las dos herramientas fue *Pet Store* [9]. Una versión completa del documento elaborado como parte de este estudio [8] puede encontrarse en http://dis.um.es/~jmolina/pfc.html.

Este artículo se organiza de la siguiente manera. Primero presentamos los criterios de comparación. En las dos secciones siguientes describimos brevemente OptimalJ y ArcStyler. A continuación presentamos el estudio comparativo y por último las conclusiones.

2 Criterios de Comparación

Durante la fase del trabajo en la que nos ocupamos de la identificación de las dimensiones y criterios de evaluación de herramientas MDA, tuvimos conocimiento de la publicación de los resultados de una interesante evaluación de la herramienta OptimalJ [2]. En dicho estudio se aplicó una metodología de análisis de características y se eligió un conjunto de propiedades extraídas de la especificación de MDA [6]. En la Tabla 1 aparece una relación de estas propiedades a la que se ha añadido P15, P16 y P17. Nuestro estudio comparativo se ha realizado a partir de estas propiedades. Por limitaciones de espacio no se describe cada propiedad, aunque entendemos que su nombre y el análisis presentado en la Sección 5 clarifican su significado.

Tabla 1. Propiedades que serán evaluadas para cada herramienta

Id	Propiedad	Id	Propiedad
P01	Soporte para PIM	P10	Soporte para consistencia incremental
P02	Soporte para PSM	P11	Soporte de transformaciones de modelos (PIM-PIM, PSM-PSM, etc.)
P03	Permite varias implementaciones	P12	Trazabilidad
P04	Integración de Modelos	P13	Soporte del ciclo de vida de desarrollo
P05	Interoperabilidad	P14	Uso de estándares
P06	Definición de transformaciones	P15	Control y refinamiento de transformaciones
P07	Verificación de modelos	P16	Calidad del código generado
P08	Expresividad de los modelos	P17	Herramientas de soporte
P09	Uso de patrones		

3 OptimalJ

OptimalJ de *Compuware* es una herramienta MDA que utiliza MOF para soportar estándares como UML y XMI. Se trata de un entorno de desarrollo que permite generar aplicaciones J2EE completas a partir de un PIM. La versión evaluada en este estudio ha sido *OptimalJ Professional Edition 3.0*, disponible a través de un convenio de colaboración de nuestro grupo con Compuware. En OptimalJ existen tres tipos de modelos [4]:

- *Modelo del Dominio*: modelo que describe el sistema a un alto nivel de abstracción, sin detalles de implementación. Corresponde al PIM de la aplicación y su elemento principal es un modelo de clases del negocio.
- *Modelo de la Aplicación*: modelo del sistema desde el punto de vista de una tecnología determinada (J2EE). Contiene los PSM de la aplicación. Se genera automáticamente a partir de un modelo del dominio y está formado por tres modelos: modelo de base de datos, modelo de interfaz web y modelo EJB.
- *Modelo de Código*: código de la aplicación, generado a partir de un modelo de la aplicación.

OptimalJ distingue varios tipos de patrones: "patrones de transformación" *entre* modelos (para transformar un PIM en PSM y un PSM en código) y "patrones funcionales" que aplican transformaciones *dentro* de un modelo (por ejemplo, *refactoring* con patrones de diseño GoF). La Figura 1 ilustra cómo se relacionan los modelos de OptimalJ y las transformaciones realizadas.

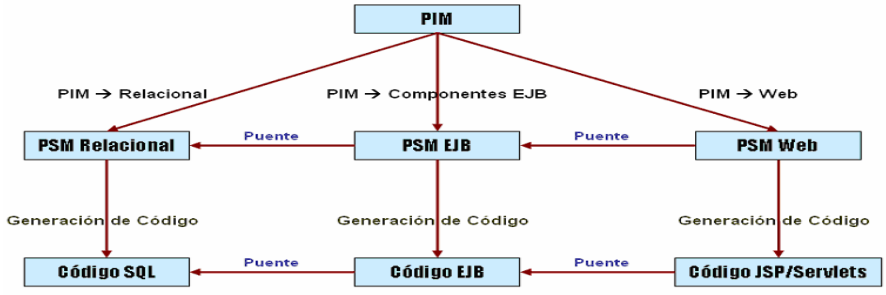

Figura 1. Conexión entre los modelos de OptimalJ

Del proceso de desarrollo con OptimalJ podemos destacar:

- Generación automática a partir del PIM de los modelos PSM de la capa de presentación (web), capa de negocio EJB y base de datos, estableciendo la conexión (puentes) entre las tres capas, como se ve en la Figura 1.
- Distinción entre *bloques libres* y *protegidos* en el código para impedir la modificación del código generado. Una nueva generación de la aplicación mantiene el código añadido en los bloques libres.
- La interfaz web generada proporciona una navegación por defecto para cada objeto de negocio, que permite el mantenimiento de los datos asociados a las clases del

Modelo del Dominio. Esa interfaz es muy pobre pero existe la posibilidad de crear un patrón de presentación que se ajuste a unas necesidades concretas, o bien manualmente modificar el código de la aplicación.

4 ArcStyler

ArcStyler de *iO-Software* es una herramienta MDA que también utiliza MOF para soportar estándares como UML y XMI, y además JMI para el acceso al repositorio de modelos. Integra herramientas de modelado (UML) y desarrollo (ingeniería inversa, explorador de modelos basado en MOF, construcción y despliegue) con la arquitectura CARAT que permite la creación, edición y mantenimiento de *cartuchos* MDA (*MDA-Cartridge*) que definen transformaciones. También incluye herramientas relacionadas con el modelado del negocio y el modelado de requisitos por lo que cubre todo el ciclo de vida. Vamos a comentar brevemente los aspectos más relevantes de ArcStyler e indicaremos cómo construir una aplicación de tres capas como la que soporta OptimalJ.

Arquitectura CARAT
Un cartucho contiene un conjunto de reglas de transformación y se instala como un *plugin*. El lenguaje de script *Jphyton* es utilizado para la creación de cartuchos. Actualmente existen numerosos cartuchos para diferentes plataformas, como J2EE, .NET, servicios web, etc. Los cartuchos utilizan *Perfiles UML* para incluir en los modelos aspectos específicos de una plataforma, por ejemplo un *perfil* para EJB o Java 2. Una funcionalidad muy potente relacionada con los cartuchos es la *herencia de cartuchos*: siendo posible definir un cartucho a partir de otro existente. A través de cartuchos podríamos definir todos los modelos de OptimalJ. En http://dis.um.es/~jmolina/documentos/CartuchosMDA.pdf se puede encontrar una introducción a la arquitectura CARAT y los cartuchos.

Marcas
En la especificación de MDA se distingue dos modos de anotar un PIM con información específica de una plataforma: Correspondencia de tipos (*Model type mapping*) y Correspondencia de instancias (*Model instance mapping*) [5]. ArcStyler proporciona los dos modos a través de estereotipos y *marcas* MDA. Las marcas son anotaciones sobre elementos de un modelo que proporcionan información a los cartuchos y que junto con los estereotipos dirigen una transformación. Un cartucho proporciona una definición del conjunto de marcas permitido para una determinada plataforma. Una marca se define con la siguiente información: tipo de dato asociado, elementos del modelo a los que se aplica y valor por defecto. Un editor inteligente de marcas evalúa las definiciones en tiempo de modelado y proporciona el soporte apropiado para marcar los modelos para una plataforma específica.

Framework Accesor
Establece un proceso bien definido para elaborar la capa de presentación de las aplicaciones, mediante la definición de un modelo *accesor* se define el flujo de control y el flujo de datos de la capa de presentación de un modo abstracto. En el proceso de generación de código, el modelo *Accessor* se implementa para una tecnología determinada. En el caso de *Pet Store* utilizamos el cartucho *WebAccesor*, que genera ficheros JSP y Java servlets. Un modelo *Accesor* incluye dos tipos de elementos:

- Un *accessor* juega el rol de un *Controlador* en la arquitectura MVC, y describe el flujo de control de una interfaz externa. Este comportamiento se modela en un diagrama de estados extendido de UML.
- Un *representer* juega el rol de una *Vista* en la arquitectura MVC. Se usa para describir un componente básico de la interfaz de usuario, por ejemplo una página JSP o ASP.NET. Se modela mediante hojas de propiedades especiales proporcionadas por ArcStyler.

En nuestra opinión, un elemento *accessor* corresponde a un *caso de uso* del sistema expresado de forma detallada y concreta. El diagrama de estados asociado a un *accessor* establece la secuencia de acciones que caracteriza a un determinado caso de uso. Esta analogía resulta muy interesante en un desarrollo dirigido por casos de uso.

Para generar una aplicación de tres capas similar a la que se ha desarrollado con Optimal-J, tendríamos que: i) utilizar el cartucho *EJB2.0* (en realidad *WSL8* que hereda de EJB); ii) tener presente que en ArcStyler no existe un modelo explícito de bases de datos sino que está implícito en el modelo EJB, de modo que *WSL8* genera una tabla de base de datos por cada componente EJB, y controla mediante marcas los aspectos de la base de datos, y iii) utilizar un cartucho *WebAccesor* con varios *accesors* y *representers* para obtener el modelo web.

5 Estudio comparativo

Tras evaluar OptimalJ y ArcStyler, mediante el desarrollo de la aplicación *PetStore* con cada una de ellas, realizamos una valoración individual que puede encontrarse en [8] y una comparación de las dos herramientas para analizar en qué medida satisfacían las propiedades de la Tabla 1. Cada una de las propiedades es valorada de 0 a 4 (nulo, mínimo, medio, bueno, excelente) según el grado en que es soportada por la herramienta. Presentamos aquí un resumen de las conclusiones obtenidas para las propiedades principales. No aparecen las propiedades *Interoperabilidad* (P05), *Soporte de estándares* (P14) y *Calidad del Código* (P16) ya que las dos primeras son claramente satisfechas, a través de XMI y el soporte de estándares como MOF, UML y el repositorio MOF, y en cuanto a la tercera, ambas herramientas generan código de calidad gracias al uso extensivo de patrones de diseño.

Soporte para PIM (P01)
Tanto OptimalJ como ArcStyler permiten crear modelos independientes de la plataforma bastante completos. En el caso de OptimalJ, el PIM del sistema está representado por el *Modelo de Clases.* Este modelo tan sólo incluye un detalle de plataforma que en nuestra opinión no debería aparecer en el PIM: la clave primaria de una clase y su restricción de unicidad asociada, que son aspectos propios del modelo de base de datos. Sin embargo, ArcStyler no muestra este detalle en el modelo, sino que establece la clave primaria de una clase mediante una *marca* en el atributo correspondiente.

En ArcStyler no hay un PIM único que dirija todo el desarrollo de la aplicación, sino que realmente tenemos dos PIM independientes: el modelo de clases que define los EJB y el modelo *Accessor* que accede a las entidades del modelo de clases. Como resultado tenemos dos PIM que están estrechamente relacionados. Esto contrasta con el PIM único de OptimalJ, punto de partida de todo el desarrollo y mediante el cual se generan el resto de modelos.

Por otra parte, ArcStyler proporciona las dos formas de anotar un PIM que se detallan en la especificación de UML a través de estereotipos y marcas. Pero al igual que en OptimalJ, un PIM de ArcStyler no está totalmente libre de detalles sobre la plataforma destino. Por ejemplo, si queremos añadir un método de búsqueda distinto al de clave primaria, debe definirse en el PIM. Conceptualmente, debería hacerse a nivel de PSM, pues representa una operación que sólo tiene sentido para la plataforma EJB. No obstante, al no disponer en ArcStyler de PSM explícitos, la única alternativa es definir estas operaciones en el PIM, enmarañándolo con detalles dependientes de la plataforma.

Soporte para PSM (P02)
El proceso típico de desarrollo con MDA contempla como pasos principales la transformación de un PIM a PSM y de uno o más PSM a código. OptimalJ sigue fielmente este esquema, generando a partir del PIM tres tipos de PSM que interactúan entre sí. Aunque estos PSM no son muy expresivos y permiten una flexibilidad limitada, nos proporcionan un modelo intermedio entre el PIM y el código, en el que podemos definir elementos específicos de una plataforma que no tienen sentido en el PIM, como métodos de búsqueda o métodos *home* para la plataforma EJB, roles de seguridad para la capa web o vistas sobre las tablas del modelo de base de datos.

ArcStyler, por contra, carece de PSM explícitos, sino que utiliza un PIM estereotipado y marcas para generar directamente el código de la aplicación para una determinada plataforma. Como vimos en el apartado anterior, esto provoca que algunos aspectos de la plataforma destino deban modelarse en el PIM.

Permite varias implementaciones (P03)
Una de las claves de MDA consiste en permitir varias plataformas de implementación a partir de un mismo PIM. En este aspecto, existe una gran diferencia entre las dos herramientas. OptimalJ permite generar tres tipos distintos de PSM relacionados entre sí y dirigidos a generar una aplicación para la plataforma J2EE. La única variación posible consiste en generar servicios web a partir del modelo EJB, e integrar la aplicación con aplicaciones externas usando CORBA o servicios web. En definitiva, se

trata de una herramienta cerrada y el soporte para nuevas plataformas lo debe proporcionar el fabricante.

En el polo opuesto está ArcStyler con la arquitectura CARAT. De partida, incorpora numerosos cartuchos MDA, cada uno de los cuales soporta una determinada plataforma (EJB, Servicios Web, .NET, JSP/Servlets, etc.). Además, es posible crear nuevos cartuchos, extendiendo alguno existente o partiendo de cero. Esta estructura abierta de ArcStyler hace que no dependa exclusivamente del soporte suministrado por el fabricante, permitiendo a una empresa adaptarse a nuevas tecnologías mediante la creación de nuevos cartuchos.

Integración de modelos (P04)

Una ventaja de OptimalJ es que genera automáticamente los puentes que conectan las tres capas de la aplicación (web, EJB y base de datos), lo cual es factible al estar orientada para una plataforma concreta. Con el código generado obtenemos una aplicación totalmente operativa en la que los JSP de la interfaz web interactúan con los EJB del modelo del negocio y estos EJB con las tablas de la base de datos, de forma totalmente transparente al desarrollador.

Sin embargo, en ArcStyler esta integración es más complicada. Al poder elegir entre tantas implementaciones posibles, un cartucho no siempre será capaz de generar los puentes adecuados para comunicarse con los elementos generados por otro cartucho, puesto que necesita conocer cómo interactuar con dichos elementos. Será responsabilidad de cada cartucho generar los puentes adecuados. En el caso de *Pet Store* hemos encontrado este problema en la comunicación entre el modelo *WebAccessor* y el modelo EJB.

Definición de transformaciones (P06)

La versión evaluada de OptimalJ no permite acceder a las definiciones de las transformaciones, que están incluidas en los llamados *patrones de transformación*. No obstante, la edición *OptimalJ Architecture Edition* permite crear nuevos patrones de transformación PIM-PSM o PSM-código.

En ArcStyler, la arquitectura CARAT permite acceder a las definiciones de las transformaciones. Desde la misma herramienta podemos extender un cartucho existente y modificarlo para satisfacer ciertos requisitos. Al igual que sucede con los patrones de transformación de OptimalJ, crear un nuevo cartucho es una tarea complicada, pero una vez construido se dispone de soporte para una nueva plataforma y puede aprovecharse para múltiples proyectos.

Verificación de modelos (P07)

OptimalJ comprueba la corrección de un modelo antes de aplicar cualquier transformación. Dispone para ello de un verificador para PIM y otro para cada tipo de PSM, cuyas comprobaciones están bien documentadas en el manual de la herramienta [4]. ArcStyler también permite usar verificadores de modelos, pero no los incorpora, sino que cada cartucho debe encargarse de implementar el verificador de sus modelos de entrada. Cabe destacar que para los cartuchos usados en nuestra evaluación (*WebAccessor* y *WLS8*) no hay documentación disponible sobre las comprobaciones que

llevan a cabo sus respectivos verificadores. Además, muchos errores pasan desapercibidos para el verificador y se trasladan a la fase de generación de código, donde resulta complicado detectar el elemento del modelo que produjo el fallo.

Expresividad de los modelos (P08)
Es muy importante que los modelos incluidos en la herramienta sean lo más expresivos posibles. De esta manera, el desarrollador tendrá mayor flexibilidad y precisión a la hora de describir las distintas partes del sistema. La importancia de esta propiedad queda patente al comparar el modelo web de ambas herramientas. Mientras que ArcStyler proporciona el modelo *Accessor* y permite diseñar interfaces gráficas a medida, en OptimalJ tenemos un modelo de presentación poco expresivo, de muy alta granularidad, que no permite especificar el flujo de control de la navegación web, imprescindible para construir aplicaciones "a medida". Modificar la navegación generada por defecto es bastante complejo y requiere cambios en el código fuente generado o bien definir nuestros propios patrones.

Uso de patrones (P09)
OptimalJ utiliza patrones de forma extensiva, tanto a nivel interno de los modelos (patrones de código, patrones de diseño, patrones del dominio) como en la generación de modelos (patrones de tecnología y patrones de implementación). Sin embargo, en ArcStyler, el uso de patrones se establece en los cartuchos. En la documentación de los cartuchos utilizados apenas se mencionan unos pocos patrones, como el *Front Controller* para los *Accessor* o el *Compact Bean* para los EJB.

Soporte para la consistencia incremental (P10)
En ambas herramientas, los cambios realizados manualmente en el código se conservan cuando se regenera toda la aplicación, mediante la distinción entre bloques libres y bloques protegidos. En el caso de OptimalJ también existe consistencia incremental sobre un PSM limitada a un pequeño conjunto de cambios. Así, algunos cambios realizados en un PSM se trasladan a otros PSM y al PIM, por ejemplo, un cambio realizado en el modelo de base de datos se refleja también en los modelos EJB y Web.

Soporte de transformaciones entre modelos (PIM-PIM, PSM-PSM, etc.) (P11)
Además de las transformaciones PIM-PSM y PSM-código las dos herramientas ofrecen un soporte limitado para otros tipos de transformaciones. OptimalJ proporciona los llamados patrones funcionales que aplican transformaciones dentro de un modelo y en el caso de ArcStyler la única transformación encontrada consiste en generar un modelo *Accessor* genérico a partir del modelo de clases, lo cual es posible gracias a los *Generic Accessors*, pertenecientes al framework *Accessor*.

Trazabilidad (P12)
OptimalJ registra todas las transformaciones realizadas por un patrón de transformación. Gracias a este registro, se puede conocer qué elemento del PIM corresponde a

un determinado elemento del PSM (clase, atributo, etc.) y su destino en el código (fichero de código, paquete, script de base de datos, etc). Sin embargo, OptimalJ no aprovecha esta información para actualizar cambios. En el caso de ArcStyler no existe tal registro.

Soporte del ciclo de vida (P13) y Herramientas de soporte (P17)
OptimalJ permite controlar prácticamente todas las fases del ciclo de vida sin necesidad de utilizar otras herramientas, pero no incluye soporte para el manejo de requisitos, sino que se parte del PIM. ArcStyler no incorpora herramientas para las fases de codificación y despliegue, ya que no dispone de un editor de código propio, y no incluye servidores de aplicaciones, ni herramientas de construcción. No obstante, gracias al mecanismo de extensibilidad de ArcStyler podemos integrar en el entorno editores de código externos como *JBuilder*. Es importante señalar, aunque no lo hemos evaluado todavía, que ArcStyler incorpora herramientas para el modelado de negocio y el modelado de requisitos como pasos previos a la construcción del PIM.

Control y refinamiento de transformaciones (P15)
El control de las transformaciones en OptimalJ es bastante limitado, y se reduce a la modificación de algunas de las propiedades disponibles en el cuadro de propiedades de determinados elementos del modelo, que pueden considerarse parámetros de transformación. Un tipo de parametrización lo encontramos en los atributos de clases del PIM, para los que es posible, por ejemplo, establecer la longitud que tendrá el dato en la base de datos o especificar una expresión en Java para el valor inicial. En ArcStyler, sin embargo, sí existe un soporte más completo para controlar y ajustar las transformaciones, gracias a las marcas asociadas a cada elemento del modelo. Una gran variedad de marcas proporciona al desarrollador una gran versatilidad a la hora de dirigir el proceso de transformación, algo que en OptimalJ no es posible.

La Tabla 2 muestra un resumen de los resultados del análisis de cada herramienta.

Tabla 2. Resumen de la evaluación de las herramientas OptimalJ y ArcStyler

Propiedad	OptimalJ	ArcStyler	Propiedad	OptimalJ	ArcStyler
P01	4	4	P10	4	3
P02	3	0	P11	2	2
P03	0	4	P12	3	1
P04	4	2	P13	3	4
P05	4	4	P14	4	4
P06	2	4	P15	2	4
P07	3	2	P16	3	3
P08	2	4	P17	3	3
P09	4	3			

6 Conclusiones y Trabajo Futuro

MDA está en su infancia y todavía queda mucho por mejorar, pero con herramientas como OptimalJ y ArcStyler podemos afirmar que MDA ya es una realidad. A continuación resumimos las conclusiones de nuestro estudio:

- Creemos que OptimalJ refleja más fielmente el proceso MDA, con una clara separación entre el PIM y los PSM, mientras que en ArcStyler el PIM se transforma directamente en código.
- Mientras que la versión evaluada de OptimalJ está orientada a la plataforma J2EE, ArcStyler es una herramienta abierta, ya que permite generar código para diferentes plataformas mediante la arquitectura CARAT. Sin embargo, debemos señalar que la versión *OptimalJ Architecture Edition* proporciona un mecanismo similar a través del lenguaje TPL.
- OptimalJ es una herramienta que en pocas horas permite crear de forma sencilla una aplicación básica para la plataforma J2EE a partir de un simple modelo de clases, generando código de calidad y aplicando patrones de diseño. Sin embargo, tiene algunas carencias que todavía necesita mejorar para permitir la construcción de aplicaciones a medida, sobre todo en relación al modelado de interfaces de usuario.
- En ArcStyler el esfuerzo de desarrollo es mayor, ya que el programador debe escribir el código de integración de modelos. Sin embargo, dispone de un buen mecanismo para el diseño de las interfaces gráficas a través del modelo *Accessor*.
- ArcStyler ofrece al desarrollador una mayor versatilidad a la hora de describir los modelos, dirigir las transformaciones y elegir distintas implementaciones.

La comparación ha puesto de manifiesto importantes decisiones de diseño para una herramienta MDA, como son la forma de anotar el PIM, que el PSM sea implícito o explícito, su naturaleza cerrada o abierta para soportar cualquier tecnología y los mecanismos para la edición y ajuste de las transformaciones.

Como trabajo futuro, nos planteamos el estudio de algunos aspectos de estas dos herramientas como: i) el lenguaje TPL de OptimalJ, ii) las capacidades de modelado de negocio y modelado de requisitos de ArcStyler y del framework *Accessor* para definir un proceso dirigido por casos de uso, y iii) un estudio más profundo de la arquitectura CARAT. También estamos evaluando otras herramientas como MDE y AndroMDA y refinando el marco de comparación para completar el trabajo realizado hasta la fecha. Por otra parte nuestro grupo ha arrancado el proyecto FraMEWeb, destinado a crear un entorno y proceso para el desarrollo dirigido por modelos de aplicaciones web.

Referencias

1. Butler Group. *Application Development Strategies*. 2003
2. Dept. Computer Science, King's College London, *An Evaluation of Compuware OptimalJ Professional Edition as a MDA Tool*. 2003.
3. *Manuales de ArcStyler Version 4.0*. Interactive Objects. 2003. www.arcstyler.com/
4. *Manuales de OptimalJ 3.0*. Compuware. 2003. www.compuware.com/products/optimalj/

5. *MDA Guide Version 1.0.1*. OMG. 2003.
6. *Model Driven Architecture (MDA)*. OMG. 2001. Documento ormsc/2001-07-01.
7. *MOF Specification*. OMG. Documento formal/02-04-03.
8. Rodríguez Vicente, J. *Ingeniería de Modelos con MDA. Estudio comparativo de OptimalJ y ArcStyler*. Proyecto fin de carrera. Facultad de Informática, Univ. de Murcia. 2004.
9. SUN Microystems, *Java Pet Store*. 2004. java.sun.com/developer/releases/petstore/