Proceso de Desarrollo de Software Mediante Herramientas MDA

Edna D. LÓPEZ L., Moisés GONZÁLEZ G., Máximo LÓPEZ S., Erick L. IDUÑATE R.
Departamento de Ciencias Computacionales
Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET)
Cuernavaca, Morelos, C.P. 62490, México
{andezepol05c, moises, maximo, lerickbe05c}@cenidet.edu.mx

RESUMEN

El enfoque Model Driven Architecture (MDA) esta centrado en los modelos y sus transformaciones, en donde para dirigir el proceso de desarrollo de software se requiere de transformaciones cada vez más detalladas. En el desarrollo de software dirigido por modelos las transformaciones de modelos son consideradas como activos importantes que deben ser manejadas con principios sólidos de ingeniería de software: estas transformaciones deben ser analizadas, diseñadas, implementadas, probadas, mantenidas y sujetas a la administración de configuración. Debido a esto existe la necesidad de identificar los modelos y sus características, técnicas y métodos que permitan el desarrollo de transformaciones y su mantenimiento. Este trabajo describe el estudio de herramientas MDA para mostrar una visión de lo que hace falta para adoptar este enfoque. Además se presentan trabajos relacionados con respecto a cada uno de los modelos de MDA y sus transformaciones, para describir el grado de definición de estos. La aportación de este trabajo es la comparativa de herramientas y trabajos relacionados, que ayuda a obtener información de lo que existe en la actualidad sobre el Model Driven Development (MDD), así como identificar las áreas de mejora y los posibles trabajos futuros que permitan complementar las características que MDA persigue.

Palabras claves: MDD**,** MDA, UML, Transformaciones, CIM, PIM, PSM.

1. INTRODUCCIÓN

El uso de modelos en disciplinas tradicionales de ingeniería es una práctica aceptada y tiene una historia amplia. Hoy en día la mayor parte de las metodologías usadas en Ingeniería de Software (IS) utilizan modelos como herramienta principal para desarrollar software debido a esto hay la tendencia a seguir un enfoque dirigido por modelos. MDD es un desarrollo de software basado en la generación de los sistemas a partir de modelos y transformaciones entre ellos. Proporciona una estrategia general a seguir en el desarrollo del software, pero no define técnicas a utilizar o fases del proceso, así como ningún tipo de guía metodológica [1].

Existen enfoques que aplican el MDD tales como: MDA; Cómputo de Modelo Integrado (MIC por sus siglas en ingles); entre otros. El enfoque de éste artículo es MDA, sus herramientas y trabajos relacionados.

MDA es una iniciativa de la Object Management Group (OMG), que representa un nuevo paradigma de desarrollo de software donde los modelos guían todo el proceso de desarrollo de acuerdo a la filosofía del MDD y se basa en estándares como UML (Unified Modeling Language), XMI (XML *Metadata Interchange*), MOF (*Meta Object Facility*) y CWM (*Common Warehouse Metamodel*). La idea clave de MDA es que si el desarrollo esta guiado por modelos de software, se obtendrán importantes beneficios en aspectos fundamentales como:

- Productividad. A través de los modelos independiente de cómputo (CIM por sus siglas en inglés), independiente de plataforma (PIM por sus siglas en inglés) y de plataforma especifica (PSM por sus siglas en inglés), se logran las transformaciones automáticamente, al menos en gran parte, al igual que la generación de código, permitiendo que el trabajo lo realice la herramienta y no el desarrollador.
- Portabilidad. Debido a que cuenta con un modelo PIM, todo lo definido en este modelo es portable hacia cualquier plataforma.
- Interoperatividad. Normalmente los modelos PSM no podrán comunicarse directamente entre ellos, ya que pueden pertenecer a distintas tecnologías. Este problema lo resuelve generando no solo los modelos PSM, sino los puentes entre ellos.
- Mantenimiento y documentación. Básicamente el modelo PIM desempeña el papel de la documentación de alto nivel que se necesita para cualquier sistema de software.

Para conseguir los beneficios mencionados, la MDA plantea el siguiente proceso de desarrollo:

- Primero los requerimientos para el sistema se presentan en un modelo CIM, que describe la situación en que el sistema será usado.
- Posteriormente este modelo es transformado en un modelo PIM que describe el sistema, pero no muestra los detalles de su uso en una plataforma tecnológica particular.
- Después de obtener el modelo PIM se realiza otra transformación hacia un modelo PSM, que contiene el detalle necesario para utilizar la plataforma tecnológica en que el sistema funcionará.
- Por último teniendo el modelo PSM se realiza una transformación que resulta en la generación de código para lograr una solución o modelo ejecutable.

MDA durante su desarrollo ha llamado la atención de investigadores de IS, así como de profesionales y proveedores de herramientas para desarrollo de software, debido tanto al auge que tiene este enfoque como los beneficios que proporciona.

Actualmente MDA cobra más fuerza en el ámbito de desarrollo de software, difundiéndose en varios eventos (como EDOC "Enterprise Distributed Object Computing", ECMDA-FA

"European Conference on Model Driven Architecture Foundations and Applications", MoDELS "Model Driven Engineering Languages and System",…) relacionados que aparecieron recientemente como se puede ver en [2]. Existe una considerable cantidad de trabajos que se están desarrollando alrededor de este enfoque y diversas herramientas que lo soportan, tanto comerciales como de *open source* que se muestran en [3].

Estas herramientas realizan transformaciones de modelo a modelo o de modelo a código, pero ninguna de ellas toman en cuenta los mismos conceptos o características, además no consideran el proceso completo de MDA, es decir, sólo toman los modelos PIM y PSM pero no CIM.

Además, cuando se está desarrollando un software mediante el enfoque MDA, es difícil determinar cuándo un modelo pasa de tener las características de un tipo de modelo a las características de otro, por ejemplo: de CIM a PIM, de PIM a PSM ó de PSM a código, ya que las transformaciones se realizan de manera gradual. La información relacionada con MDA tiene diversas representaciones, no tiene bien definido como se deben de llevar acabo las transformaciones ni que información debe contener cada uno de sus modelos. Debido a eso es necesario tener un marco de referencia del estado del arte de MDA con respecto a los trabajos y herramientas derivados alrededor de este enfoque para saber los aspectos a embestir en la actualidad.

Este artículo se organiza de la siguiente manera: la sección 2 muestra el análisis y descripción de trabajos relacionados a la definición de los modelos de MDA y sus transformaciones con el propósito de identificar el avance alcanzado. Posteriormente en la sección 3 se analizan las herramientas con enfoque de MDA que sobresalen en la actualidad, sus modelos y transformaciones para determinar los avances y áreas de mejora. Por último, en la sección 4 se muestran las conclusiones y posibles trabajos futuro.

2. COBERTURA DE MODELOS DE MDA

Con el objetivo de revisar el avance y áreas de mejora que se tiene en la actualidad en los diversos trabajos en torno al enfoque MDA considerando la delimitación de cada una de sus fronteras (CIM, PIM y PSM) y transformaciones, es necesario realizar este estudio. A continuación se describen brevemente dichos trabajos y en la tabla 1 se ilustra una comparativa entre ellos resaltando las características deseables para soportar MDA.

2.1. Separación de aspectos en MDA: Una aproximación basada en múltiples vistas [4]

Propone integrar técnicas de múltiples perspectivas (se basan en percibir un sistema de software desde diferentes ángulos/posiciones, por ejemplo CIM, PIM y PSM) con MDA y separación de aspectos multidimencionales (dimensión de clases y otra de casos de uso) en cada una de dichas perspectivas. La representación de los modelos que utiliza es con los estereotipos de UML para especificar las vistas de CIM, PIM y PSM. Los objetivos que cubre son: lograr una separación de aspectos (lógica del negocio, requerimientos,…) en el nivel CIM (se utilizan casos de uso) y eliminar los Crosscutting[1] Concerns que aparecen dentro de cada punto de vista que modela el sistema.

2.2. Ingeniería dirigida por modelos reflexivos [5]

Propone aplicar el enfoque MDA a si mismo, con las nociones de PIT (Transformaciones Independientes de Plataforma) y PST (Transformaciones de Plataforma Especifica) que dan a MDA dos pasos nuevos en el ciclo de vida de transformaciones de modelos, el primero expresa las transformaciones de modelo de una manera independiente del instrumento (herramienta) y el segundo marca esta expresión independiente del instrumento sobre un instrumento actual. En este artículo, cuando se habla de plataformas son entendidas como los instrumentos que permiten la especificación, el diseño y la ejecución de transformaciones. Con el objetivo de transformar de PIM-PIM, PIM-PSM y PSM-PSM se utilizan PIT y PST. Se basa en el enfoque orientado a objetos y toma a UML como un metamodelo que define lenguajes de transformación de tipo PIT y PST.

2.3. Una propuesta de proceso explícito de V&V en el marco de MDA [6]

Presenta un proceso V&V (Validación & Verificación) genérico para ser aplicado en el ámbito de metodologías basadas en MDA o MDD usado para detectar errores e inconsistencias en las etapas tempranas del desarrollo, evitando así la propagación de esos errores a las etapas posteriores, ya que un error en el PIM se arrastra a un PSM hasta llegar al código. Además el proceso sugerido sirve para identificar funciones importantes que deberían cumplir las herramientas de soporte de MDA respecto a realizar modelos correctos por medio de la verificación y validación. Para pasar al modelado del PSM el proceso de verificación y validación debe ser realizado tanto para el análisis de requisitos como para el modelado del PIM.

2.4. AspectMDA: Hacia un desarrollo incremental consistente integrando MDA y orientación a aspectos [7]

Se basa en la idea de especificar diferentes modelos que corresponden con distintos aspectos del sistema (como seguridad, restricciones de tiempo real,…), este integra a MDA, DSOA[2] (Desarrollo de Software Orientado a Aspectos) y xlinkit[3] de una forma adecuada y beneficiosa para el MDD. Los objetivos de este artículo son: modelar esos aspectos (artefactos o propiedades de un sistema) por cada equipo de trabajo desde el CIM hasta el PSM con el mínimo de comunicación, verificar la consistencia, mejorar la trazabilidad entre los diferentes niveles de abstracción y controlar el impacto al cambio. De este modo, los niveles MDA se especifican en diferentes facetas del sistema que modela los aspectos y los mantienen separados durante todo el marco (MDA).

2.5. Definición y descripción de PIM [8]

Presenta un enfoque de Ingeniería Dirigida por Modelos (MDE por sus siglas en inglés), que utiliza la visión MDA. El MDE toma en consideración dos aspectos importantes: la separación

[1] Crosscutting Concerns, es un problema que se presenta cuando aparecen aspectos en el sistema que no pueden ser modelados de forma independiente a otros aspectos.

[2] Modela los componentes y aspectos como dos entidades separadas donde los aspectos se mezclan o componen de forma automática con el comportamiento funcional del sistema.

[3] Es una propuesta para administrar la consistencia de documentos XML heterogéneos distribuidos por la red que son centrales para el desarrollo de sistemas de software.

de aspectos funcionales (ejemplo requerimientos) de los no funcionales (ejemplo calidad) de un sistema y los aspectos de plataforma independiente de los de plataforma específica. Se definen las partes que conforman a un PIM como: Contexto (el alcance del sistema a ser desarrollado), Requerimientos, Análisis (especifica la vista interna del sistema) y Diseño del componente (solución independiente de plataforma expresada en términos de componentes de software).

2.6. Conceptos y técnicas de PSM [9] y Técnicas de mapeo de PIM a PSM [10]

Tanto [9] como [10] son proyectos de MASTER (Model-driven Architecture inSTrumentation, Enhancement and Refinement) que es el nombre de un proyecto cuyo objetivo es realizar una revisión de los conceptos de MDA. En [9] se mencionan las características que debe tener un PSM para ser construido de acuerdo al perfil de la plataforma, librerías, conjunto de reglas, patrones y metamodelos. En [10] se centra en las transformaciones de PIM a PSM que deben considerar las características de trazabilidad, bidireccionalidad, consistencia y simplicidad (para facilitar las pruebas o verificación en la transformación). Utilizan separación de aspectos para el mapeo de PIM a PSM, refiriéndose a que el PSM aborda tanto cuestiones tecnológicas como de negocios, por lo que se deben separar y a esta separación le llama separación de aspectos, para que la parte del negocio no se disperse en la tecnología, sino que el modelo PIM del negocio se enriquezca de la tecnología.

2.7. Enfoque basado en componente orientado a características para la transformación de CIM a PIM [11]

Presenta un enfoque de transformaciones de CIM a PIM, considerando al CIM como un modelo de características (características) y al PIM como la arquitectura de software (componentes). Se resuelven parcialmente 2 problemas con respecto a la transformación de CIM a PIM: el seguimiento de CIM a PIM (como los elementos en el CIM pueden ser remontados a elementos en el modelo PIM) y la construcción de PIM basado en CIM (como formar los elementos del PIM en la transformación).

Con el fin de bosquejar las características que presentan los trabajos mencionados con respecto al soporte de MDA, en la tabla 1 se presentan las características a evaluar (columnas) y los autores de los trabajos referenciados (renglones). La intersección de un renglón con una columna se representa con el símbolo "√" ó "*". A continuación se describen las columnas y símbolos de la tabla 1.

Columnas:

- Diagramas UML. Utiliza diagramas UML para el modelado.
- Consistencia. Es importante para mantener la información adicional que se requiere para transformar un modelo en otro, de manera que pueda reutilizarse cuando se requiera una modificación o actualización del modelo fuente para volver a realizar su transformación al modelo objetivo.
- Trazabilidad. Implica conocer el elemento origen a partir del cual se ha generado cualquier elemento del modelo destino.
- Especificación. Define la información requerida del modelo, para CIM, PIM, PSM o todos.
- Transformaciones. Soporte a transformaciones entre diferentes niveles de abstracción como: CIM-PIM, PIM-PSM y de PSM-IM (Modelo de Implementación).

Símbolos:

- La "√" significa que cumple la característica al 100%.
- El "*" indican que cumple con la característica pero no al 100%, es decir, no especifica de manera detallada lo que contiene el modelo (ejemplo, CIM).

En la comparativa de la tabla 1 todos los trabajos contemplan la utilización de UML para el modelado, excepto [11]. Las dos características deseables como consistencia y trazabilidad solamente [7] las considera, además en [7] se realizan los 3 tipos de transformaciones, aunque no defina cada modelo. Ninguno de los trabajos abarca la especificación de los 3 modelos MDA, ni sus 3 transformaciones (CIM-PIM, PIM-PSM, PSM-IM). Lo que significa que hace falta mejorar en la definición y descripción de los modelos y sus transformaciones, para encontrar una manera homogénea de representación y entendimiento.

Tabla 1. Comparativa de trabajos relacionados

Artículos	Diagramas UML	Consistencia	Trazabilidad	Especificación			Transformación		
				CIM	PIM	PSM	CIM-PIM	PIM-PSM	PSM-IM
[4]	√		√	*			√		
[5]	√							√	
[6]	√	√			*				
[7]	√	√	√				√	√	√
[8]	√				√				
[9] y [10]	√					√		√	
[11]			√				√		

3. HERRAMIENTAS MDA

Debido al auge de MDA se han desarrollado herramientas tanto *open source* como comerciales que dan soporte parcial o total al enfoque, en [2] y [3] se muestran dichas herramientas. El soporte para MDA puede darse en diversas formas, como en la generación de código a partir de modelos y en las transformaciones de modelos, donde ambas pueden ser implementadas por distintas herramientas: herramientas de transformación de PIM a PSM, PSM a código, PIM a código, ajustables y de definición de transformaciones. Las herramientas analizadas que se destacan debido a su madures, uso y mayor soporte al enfoque son: ArcStyler, OptimalJ, AndroMDA, Codagen Architect y Together Architect. En la tabla 2 se ilustra una comparativa entre dichas herramientas. Los tipos de herramientas presentadas son de transformaciones de modelo a modelo (PIM a PSM) y de modelo a código (PIM a código, PSM a código).

El estudio de estas herramientas permite conocer las características de una herramienta MDA, sus limitaciones actuales y su aplicabilidad, mostrando un panorama de lo que falta mejorar e implementar en ellas respecto a las transformaciones, definiciones de sus modelos, forma de construcción y tecnologías (Ej. UML, XMI, MOF,…) que utilizan.

En la tabla 2 se muestran las características que especifican las herramientas con respecto al soporte de MDA para identificar en que medida se cubren. Se presentan las características a evaluar (renglones) y los nombres de las herramientas (columnas). La intersección de un renglón con una columna es

representada con el símbolo "√" ó "*". A continuación se describen los renglones y símbolos de la tabla 2.

Renglones:

- Entrada. Tipo de entrada que admite para desarrollar la aplicación.
- Salida. Tipo de lenguaje o plataforma en la cuál genera el código.
- Estándares. Estándares en que se basa la herramienta.
- Soporte de CIM. Creación de modelos independientes de cómputo.
- Soporte PIM. Creación de modelos independientes de plataforma.
- Soporte PSM. Creación de modelos de plataforma específica.
- CIM a PIM. Soporte a transformaciones del modelo independiente de cómputo al modelo independiente de plataforma.
- PIM a PSM. Soporte a transformaciones del modelo independiente de plataforma al modelo de plataforma específico.
- PSM a IM. Soporte a transformaciones del modelo de plataforma específica al modelo de implementación (Generación de código).
- Consistencia. Es importante para mantener la información adicional que se requiere para transformar un modelo en otro, de manera que pueda reutilizarse cuando se requiera una modificación o actualización del modelo fuente para volver a realizar su transformación al modelo objetivo.
- Trazabilidad. Implica conocer el elemento origen a partir del cual se ha generado cualquier elemento del modelo destino.

Símbolos:

- La "√" significa que cumple la característica al 100%.
- El "*" significa que cumple la característica pero indirectamente.

Cuando se menciona "indirectamente" significa que implícitamente cumple la característica.

Los diagramas UML que soportan las herramientas es importante conocerlos, debido a que de ello se podría deducir cuales son los diagramas más utilizados para el desarrollo de aplicaciones a través del enfoque MDA. Los diagramas de UML que soportan estas herramientas son: casos de uso, clases, secuencia, colaboración, actividad, máquina de estado, componentes, objetos, cronometraje, visión general de interacción, estructura compuesta, así como el diagrama de distribución. La herramienta OptimalJ utiliza todos excepto los diagramas de cronometraje, visión general de interacción y estructura de interacción. Arcstyler no utiliza el diagrama de objetos, cronometraje, visión general de interacción y estructura de interacción. Together Architect no utiliza los diagramas de máquina de estados, objetos, cronometraje, estructura compuesta. Codagen Architect y AndroMDA utilizan los diagramas de clases, casos de uso y estados. Además AndroMDA usa el diagrama de secuencias y Codagen Architect el diagrama de actividad. Los resultados de la tabla 2 se puntualizan a continuación:

- La mayoría de las herramientas no implementan en su totalidad la especificación completa de MDA, estas pueden realizar las transformaciones entre los modelos PIM-PSM, PSM-Código, pero no de CIM-PIM o simplemente realizan las transformaciones directas de PIM a código (por ejemplo ArcStyler, Codagen Architect).
- Ninguna de las herramientas tiene soporte a la definición de todos los modelos de MDA, solamente soportan a PIM y PSM y Together Architect solamente considera al CIM indirectamente.
- La consistencia algunas herramientas la especifican y otras ligeramente la tratan, además de que es un punto relevante para el enfoque MDA con respecto a las transformaciones. Las herramientas que cumplen esta característica son ArcStyler, OptimalJ, Codagen Architect, las demás herramientas tienen un soporte ligero a esta característica.
- La trazabilidad es un punto relevante y que sólo algunas herramientas la consideran con mayor empeño, ya que es significativo para la búsqueda y corrección de errores y es uno de los puntos deseable para las transformaciones. Las herramientas que especifican esta característica son OptimalJ, ArcStyler.
- Debido a que la base del estándar MDA son UML, MOF y XMI, es importante que las herramientas consideren estos estándares para encontrar uniformidad en la interpretación del enfoque MDA. Las herramientas analizadas que consideran estos estándares son ArcStyler y OptimalJ.

Tabla 2. Comparación de herramientas MDA [12, 13, 14**]**

Herramientas	*ArcStyler [15]*	*OptimalJ [16]*	*AndroMDA [17, 18]*	*Codagen Architect*	*Together Architect*
Entrada	UML	XMI v1.1	XMI v1.1	XMI v1.1, MDD de Racional, Visio, together.	XMI
Lenguaje de salida	Todos	En J2EE	Todos	Java,C#,C++, Visual Basic.	Java,C#,C++,Corba, Visual Basic 6,Visual Basic .Net
Estándares	XMI, UML, MOF, JMI	XMI, UML, MOF, XML, WSDL, J2EE	Struct, Netbeans MDR, antMybernate, Velocity, xDoclet y Maven.	Struct	XMI
Soporte CIM					*
Soporte PIM	√	√	√	*	*
Soporte PSM	*	√	√	*	√
CIM a PIM					
PIM a PSM		√	√		
PIM a Código	√			√	
PSM a Código		√	√		
Consistencia	√	√	*	√	*
Trazabilidad	√	√	*	*	*

4. CONCLUSIONES

Aunque existen varias herramientas y trabajos relacionados que aplican el enfoque MDA, muchas de estas no lo contemplan en su totalidad y cada una tiene una forma diferente de interpretar

cada modelo. Con el análisis realizado se observa que hace falta mejorar en la definición y delimitación de cada uno de los modelos de MDA, así como la determinación de sus transformaciones. Buscando obtener representaciones homogéneas (ejemplo definición de la información de CIM, PIM, PSM) en la forma de aplicar el enfoque.

De las herramientas MDA que presentan mejores características para el soporte de este enfoque son: ArcStyler, AndroMDA y OptimalJ ya que implementan el enfoque MDA casi en su totalidad. OptimalJ es la herramienta que mejor cumple el enfoque MDA, al realizar transformaciones entre modelos PIM, PSM y generar código. Además esta herramienta considera en buena medida la trazabilidad y consistencia igual que ArcStyler ya que pocas herramientas especifican estas características y no en su totalidad. En todas las herramientas analizadas existe la carencia de la especificación del CIM, así como su transformación hacia el PSM.

En futuros trabajos es necesaria una caracterización más completa de las herramientas y trabajos alrededor del enfoque MDA, para lograr la aplicación normalizada del enfoque e identificar la información necesaria en cada uno de sus modelos y definición de las transformaciones.

5. REFERENCIAS

[1] Bran Selic,"Model-Driven Development: Its Essence and Opportunities",Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, 2006.

[2] DSDM: Desarrollo de Software Dirigido por Modelos. MDA y Aplicaciones, http://www.lcc.uma.es/~av/MDD-MDA/, revisada agosto de 2006.

[3] OMG, Committed Companies and Their Products, http://www.omg.org/mda/committed-products.htm, revisada agosto de 2006.

[4] Pablo Amaya, Carlos González, Juan M. Murillo, "Separación de Aspectos en MDA: Una aproximación basada en múltiples vistas", Actas del I Taller sobre Desarrollo Dirigido por Modelos, MDA y Aplicaciones (DSDM'04), Málaga, España, 9 de noviembre de 2004.

[5] Jean Bézivin, Nicolas Farcet, Jean-Marc Jézéque, Benoıt Langlois, and Damien Pollet, "Reflective Model Driven Engineering", Proceedings of UML 2003, San Francisco, volume 2863 of LNCS, pag. 175-189, Springer (10/2003).

[6] Francisco Javier Lucas Martínez, Fernando Molina Molina, Ambrosio Toval Álvarez, "Una Propuesta de Proceso Explícito de V&V en el Marco de MDA", Actas del II Taller sobre Desarrollo Dirigido por Modelos, MDA y Aplicaciones (DSDM'05), Málaga, España, 13 de septiembre de 2005.

[7] Pablo Amaya Barbosa, Carlos González y Juan M. Murillo Rodríguez, "AspectMDA: Hacia un desarrollo incremental consistente integrando MDA y Orientado a Aspectos", Actas del II Taller sobre Desarrollo Dirigido por Modelos, MDA y Aplicaciones (DSDM'05), Málaga, España, 13 de septiembre de 2005.

[8] Daniel Exertier, Benoit Langlois Xavier Le Roux, "PIM Definition and Description",Proceedings First European Workshop on Model Driven Architecture with Emphasis on Inductrial Application, Univ. Twente, Netherlands , pag. 17-18, Marzo 2004.

[9] Model-driven Architecture inSTrumentation, Enhancement and Refinement (MASTER),"PIM to PSM mapping techniques", http://modeldrivenarchitecture.esi.es/mda_public Documents.html#D2.1, Diciembre 2003.

[10] Model-driven Architecture inSTrumentation, Enhancement and Refinement (MASTER),"PSMs Concepts and Techniques", http://modeldrivenarchitecture.esi.es/mda_publicDocuments.html#D2.1 Junio 2003.

[11] Wei Zhang, Hong Mei, Haiyan Zhao, Jie Yang, "Transformation from CIM to PIM: A Feature-Oriented Component-Based Approach", Models 2005, Jamaica, volume 3713 of LNCS, pag. 248-263, 2005.

[12] Naveed Ahsan Tariq, Naeem Akhter, "Comparison of Model Driven Architecture (MDA) based tools (A Thesis document)", http://dis.dsv.su.se/~emisnat/CMDA/MDATools_ KTH_KUH_MasterThesis2004_ Telemed5.pdf, 2004, Revisada septiembre de 2006.

[13] J. García Molina, J. Rodríguez, M. Menárguez, M.J. Ortín, J. Sánchez, "Un estudio comparativo de dos herramientas MDA: OptimalJ y ArcStyler", Actas del I Taller sobre Desarrollo Dirigido por Modelos, MDA y Aplicaciones (DSDM'04), Málaga, España, 9 de noviembre de 2004.

[14] Luis Enrique Corredera de Colsa, "Arquitectura dirigida por modelos para J2ME", http://personal.telefonica.terra.es /web/lencorredera/mda_j2me.pdf, Revisada septiembre de 2006.

[15] ArcStyle, http://www.arcstyler.com/, Revisada el septiembre de 2006.

[16] OptimalJ, http://www.compuware.com/products/optimalj/, Revisada septiembre de 2006.

[17] AndroMDA, www.andromda.org2, Revisada septiembre de 2006.

[18] Chris Micali, "Introduction to Model Driven Development with AndroMDA (Part 1)", http://www.codeproject.com/ useritems/intro_to_andromda_1.asp, Revisada septiembre de 2006.