PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Reporte Técnico RT 07-02

Extensión MDA (Model Driven Architecture para proceso basado en RUP (Rational Unified Process)

Andrea Delgado Natacha Carballal Catalina Rapetti

2007

Extensión MDA (Model Driven Architecture para proceso basado en RUP (Rational Unified Process)
Delgado, Andrea; Carballal, Natacha; Rapetti, Catalina
ISSN 0797-6410
Reporte Técnico RT 07-02
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay, 2007

Extensión MDA (Model Driven Architecture) para proceso basado en RUP (Rational Unified Process) ϕ

Andrea Delgado, Natacha Carballal, Catalina Rapetti
Universidad de la República,
Instituto de Computación,
Grupo de Ingeniería de Software,
Montevideo, Uruguay
{adelgado, pgmdapis}@fing.edu.uy

Febrero 2007

Resumen

El enfoque de desarrollo Model Driven Architecture (MDA) propone basar el desarrollo en modelos, separando la especificación del sistema de las plataformas utilizadas. El Grupo de Ingeniería de Software (Gris) del Instituto de Computación tiene como eje de sus actividades un programa de construcción y prueba de modelos de proceso en el curso "Proyecto de Ingeniería de Software" desde el año 2000.

El objetivo de este trabajo en progreso es realizar y probar una Extensión MDA al proceso basado en RUP que se tiene definido, modificando principalmente las actividades y entregables asociados a las Disciplinas de Diseño e Implementación, y seleccionando una herramienta con soporte MDA. Se presenta el enfoque MDA, el proceso basado en RUP y el contexto y prueba de la Extensión MDA definida.

Palabras clave: Ingeniería de Software, Procesos y Metodologías de desarrollo, Arquitecturas de Software, Model Driven Architecture (MDA)

ϕ Artículo publicado en las "VI Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC'07)", Lima, Perú, Febrero de 2007. ISBN 978-9972-2885-1-7

Introducción

El grupo de Ingeniería de Software (Gris) del Instituto de Computación de la Facultad de Ingeniería de la Universidad de la República, Uruguay [1] tiene como eje principal de sus actividades un programa de construcción y prueba de modelos de proceso, que inició en el año 2000 y utiliza como banco de pruebas de los procesos al curso "Proyecto de Ingeniería de Software" [2] que se dicta en cuarto año de la carrera de Ingeniería en Computación.

El objetivo central del programa es el desarrollo de modelos de proceso que puedan resultar adecuados para su transferencia a la industria, para lo que, entre otras características, se cuenta con clientes externos al curso. Estos clientes plantean el desarrollo de sistemas de mediano porte que en general presentan desafíos tecnológicos importantes y son en general grupos de investigación, proyectos o áreas de la misma facultad, o empresas del medio. Desde su puesta en marcha y hasta la fecha se han estudiado y adaptado varios modelos de proceso de las distintas corrientes existentes, tanto procesos "pesados" como "ágiles", siendo el representante más importante de los primeros el Rational Unified Process (RUP) [3] y de los segundos el eXtreme Programming (XP) [4], si bien existen "agilizaciones" del RUP que no se han abordado. De los dos procesos mencionados se han realizado adaptaciones que se han puesto a prueba en distintas ediciones del curso, definiéndose como proceso base del programa la adaptación realizada del RUP que es sobre la cual se agregan otros enfoques y metodologías de desarrollo cuya descripción puede verse en [5].

La metodología para el desarrollo de aplicaciones con enfoque MDA se inscribe en este programa de prueba de procesos y está siendo desarrollada por parte de un proyecto de fin de carrera de quinto año de la carrera de Ingeniería en Computación, dirigido por un docente del Grupo de Ingeniería de Software (Gris). La definición de la Extensión MDA [6] se basa en el estándar del Object Management Group (OMG) [7], agregando actividades y entregables específicos al proceso base adaptación del RUP, y la utilización de una herramienta que cumpla con el enfoque MDA y que mejor se adapte a las características del curso.

1 Model Driven Architecture (MDA)

El enfoque de desarrollo Model Driven Architecture (MDA) según la especificación en [7] se basa en la utilización de modelos en el desarrollo de software, y en la idea de separar la especificación de un sistema, de los detalles de la forma en que dicho sistema utiliza las capacidades de sus plataformas. Está guiado por modelos en tanto provee sentido a la utilización de modelos como base para dirigir el curso de la comprensión, diseño, construcción, deployment, operación, mantenimiento y modificación de los sistemas. A partir de la separación de intereses planteada en el enfoque, se buscan obtener tres metas principales para los sistemas desarrollados: portabilidad, interoperabilidad y reusabilidad. Esta separación de intereses se ve reflejada en la definición de las tres vistas de un sistema que se proveen en los modelos definidos: Computation Independent Model (CIM), Platform Independent Model (PIM) y Platform Specific Model (PSM).

El ciclo de vida en un desarrollo utilizando MDA comienza como en cualquier proceso de desarrollo por la captura de requerimientos. Estos en su mayoría se especifican en forma de texto, sobre los cuales se realiza el análisis de requerimientos que da lugar al primer modelo del sistema que es un modelo conceptual del mismo. Este modelo en el enfoque MDA se conoce como CIM (Computation Independent Model) para el cual en general no se definen transformaciones. A partir del CIM en forma manual se realiza el PIM (Platform Independent Model) que es un modelo de diseño independiente de la implementación, sobre el cual se definen transformaciones que llevarán ese modelo al PSM (Platform Specific Model) a partir del cual se generará el código correspondiente a la plataforma elegida. Es importante notar que a partir de un mismo PIM se pueden generar tantos PSM como plataformas objetivo se definan, y también se puede generar directamente el código sin pasar por un PSM. El aspecto central del enfoque se encuentra en la transformación de modelos, que según la definición provista por el estándar, es el proceso de convertir un modelo en otro modelo del mismo sistema. En la figura 1 se presenta la secuencia general de pasos definidos en la aplicación del enfoque según [8]:

Figura 1. Los tres pasos principales del enfoque MDA

2 Conceptos en MDA

En esta sección se describen los principales conceptos que definen el enfoque MDA según [7]. El Computation Independent Model (CIM) es una vista del sistema desde el punto de vista independiente de la computación. Puede estar compuesto por varios artefactos como Modelo de Dominio, especificación de requerimientos como Modelo de Casos de Uso, y es independiente de cómo el sistema es implementado. Actúa como entrada para el resto de los modelos y como fuente de vocabulario compartido para usar en otros modelos. Los requerimientos establecidos en el CIM deben ser trazables al PIM y al PSM que los implementan, y viceversa.

El Platform Independent Model (PIM) es una vista del sistema desde el punto de vista del diseño independiente de la plataforma. Describe el sistema desde el punto de vista del software que lo compone, sin presentar detalles de cómo serán utilizadas las plataformas asociadas. En base a dicha independencia se puede utilizar en la generación de modelos específicos para varias plataformas distintas. Por ejemplo, se puede asumir la disponibilidad de características de tipo general de plataforma, como ser invocación remota, sin especificar más que eso, luego la característica indicada será implementada según la plataforma.

El Platform Specific Model (PSM) es una vista del sistema desde el punto de vista específico de la plataforma. Un PSM combina las especificaciones en el PIM, con los detalles que definen como el sistema utiliza ese tipo particular de plataforma. El PSM puede ser el código para la plataforma elegida, si provee toda la información necesaria para construir el sistema y ponerlo en operación, o puede actuar como un PIM que se utilizará para refinamientos posteriores en un PSM que pueda ser implementado directamente.

Los Mapeos entre modelos proveen las especificaciones para realizar las transformaciones de un PIM en un PSM para una plataforma específica. El modelo

asociado a la plataforma, determinará la naturaleza de los mapeos. Por ejemplo, un PIM especificado en UML mapeado a EJB provee marcas que serán utilizadas para guiar la transformación del PIM al PSM, de forma que marcando una clase con la marca de “session”, resulta en la transformación de esa clase según el mapeo establecido, en un “session bean” y otras clases soporte. Un mapeo es especificado utilizando algún lenguaje para describir una transformación de un modelo en otro. La descripción puede ser en lenguaje natural, un algoritmo en un lenguaje de acciones según lo establecido en el Precise Action Semantics incorporado a UML 2.0, o en un lenguaje de mapeo de modelos como se define en el estándar QVT. Para realizar el mapeo de modelos se definen dos enfoques: mapeo de tipos y de instancias.

El Marcado de modelos se utiliza para los mapeos, dichas marcas provienen de distintas fuentes: tipos de un modelo (clases, asociaciones, etc.), roles de un modelo (patrones), estereotipos de un perfil UML, elementos de un modelo MOF, elementos de modelos especificados por cualquier metamodelo. Para que las marcas puedan ser utilizadas adecuadamente, deben ser estructuradas, restringidas y/o modeladas, por ejemplo un conjunto de marcas que indique alternativas mutuamente excluyentes para un concepto, deben ser agrupadas de forma que al marcar el modelo se sepa cuales son las posibles elecciones y que más de una de esas marcas no pueden ser aplicadas al mismo elemento del modelo. Algunas marcas incluso podrían necesitar parámetros, como las de QoS, por ejemplo “soportar conexiones simultáneas” podría requerir un parámetro que indique cota superior de la cantidad de conexiones que debe soportar, o incluso varios parámetros indicando detalles para time-outs o políticas de conexión. Un conjunto de marcas podría ser especificado por un modelo de marcas independiente de los mapeos, en vez de ser provisto por un mapeo en particular, y así ser usado por distintos mapeos, o ser provisto con un perfil UML.

La Transformación entre modelos es la base del enfoque, luego que se tiene el PIM marcado, el siguiente paso es transformarlo en un PSM. Esta transformación puede ser realizada según distintos enfoques: en forma manual, asistida por computadora o en forma automática. La entrada para la transformación es el PIM marcado y el mapeo, el resultado de la misma es el PSM obtenido o el código generado y el registro de la transformación. La transformación es realizada según el tipo de mapeo que se utilice: de tipos o instancias en modelos. Los resultados de la transformación de un PIM utilizando una técnica particular, son el PSM obtenido en la transformación y el registro de la misma. Este registro contendrá un mapeo del elemento en el PIM a los correspondientes elementos en el PSM, mostrando que parte del mapeo fue utilizado en cada parte de la transformación. Una herramienta que lleve un registro de la transformación podría sincronizar el PIM y el PSM generado, cuando alguno sea modificado, y con el código obtenido.

3 Modelado y estándares en OMG

El Object Management Group (OMG) utiliza una arquitectura de cuatro capas de modelado sobre la cual está basada la definición de sus estándares, y el concepto de metamodelado definido como: el uso de modelos (en una capa de abstracción) para describir modelos (en una capa inferior siguiente). Estas capas se notan M0, M1, M2 y M3; cada una y sus dependencias están definidas por el OMG y especificadas en [7] como sigue:

Capa M3, el Meta-Meta-Modelo: contiene entidades para definir lenguajes de modelado. El estándar Meta-Object Facility (MOF) que se ubica en esta capa, es el lenguaje para definir lenguajes de modelado, por ejemplo una “MOF Class”. Los elementos de la capa M3 se definen con instancias de conceptos de esta capa, o sea que MOF se define en base a MOF.

Capa M2, el Meta-Modelo: en esta capa se ubican los metamodelos que contienen las entidades de los lenguajes de modelado definidos. Los estándares Unified Modeling Language (UML) y Common Warehouse Metamodel (CWM) están en esta capa, y definen los elementos con los cuales modelar sistemas, por ejemplo “UML Class”. Los elementos en esta capa son instancias de los elementos de la capa M3.

Capa M1, el Modelo: contiene las entidades con que se modela un sistema en particular, según el lenguaje de modelado utilizado, por ejemplo en UML podría ser la Clase Socio con Atributos Nombre y Apellido. Los elementos en esta capa son instancias de los elementos en la capa M2.
Capa M0, los Datos: contiene los datos específicos que maneja el sistema modelado según la capa M1, por ejemplo el Socio “Juan Perez”, que es una instancia de la Clase Socio con Atributos Nombre y Apellido definidos en la capa M1. Los elementos en esta capa son instancias de los elementos en la capa M1.

Además define una variedad de estándares sobre algunos de los cuales se basa el enfoque MDA. La importancia de los mismos radica en que la utilización de estos estándares permite realizar en forma automática, validación y ejecución de modelos, y transformación de modelos, lo que como ya se mencionó, constituye el elemento central del enfoque.

Meta Object Facility (MOF): establece un lenguaje común para definir lenguajes de modelado como UML, lo que permitirá definir transformaciones sobre los metamodelos en forma estándar. En base a MOF se define un repositorio de modelos que puede utilizarse para especificar y manipular modelos.

XML Metadata Interchange (XMI): permite el intercambio de modelos via documentos XML. XMI se utiliza para intercambiar modelos en el nivel M1, y para generar formatos de intercambio de metamodelos en la capa M2.

Unified Modeling Language (UML): lenguaje de modelado estándar para especificar, visualizar y documentar sistemas de software. En UML 2 se integran un conjunto de conceptos para especificar completamente el comportamiento de objetos, el UML Precise Action Semantics, que permite agregar dicha información.

Perfil UML: mecanismo de extensión de UML. Se aplica a la especificación de un lenguaje, especificando un nuevo lenguaje de modelado mediante el agregado de nuevos elementos o restricciones al lenguaje. Varios perfiles pueden aplicarse a un modelo, extendiendo o restringiendo los elementos de ese modelo. Actualmente existen perfiles para CORBA, Java, EJB y C++, lo que permitirá construir PSMs específicos para estas tecnologías.

Object Constraint Language (OCL): lenguaje de especificación para escribir expresiones sobre modelos, por ejemplo invariantes, pre y postcondiciones, y cuerpo

de operaciones. Mediante OCL se podrían especificar transformaciones describiendo los elementos en el modelo origen y destino.

Query, Views and Transformations (QVT): define lenguajes usando MOF para especificar como se realizan las transformaciones entre modelos. Es un estándar con gran relevancia para MDA, ya que establece un modo estándar de definir transformaciones entre modelos. Se definen lenguajes para crear vistas de un modelo, realizar consultas sobre modelos y escribir definiciones de transformaciones entre modelos, siendo este último el de mayor relevancia para MDA, el Model Transformation Language (MLT) que utiliza el "pattern matching" como uno de sus factores clave para permitir la definición de transformaciones entre modelos.

4 Proceso y metodología de desarrollo

Muchos procesos de desarrollo de software en la actualidad tienen en cuenta que los requerimientos de los sistemas son inestables y debe ser posible incorporar cambios en los mismos durante el desarrollo a medida que van surgiendo, así como agregar nuevos requerimientos. Estos modelos, ya sean "pesados" o "ágiles", siguen la filosofía de la interacción y el cambio, e indican liberaciones frecuentes que son utilizadas para obtener feedback de los usuarios, y la construcción del sistema en forma iterativa incremental sobre estas liberaciones.

Una metodología para el desarrollo basado en MDA incorpora la concepción de que los modelos que se realizan en las distintas etapas, no son meramente documentación del sistema para utilizar durante las distintas etapas del desarrollo y posterior mantenimiento de la aplicación, sino que constituyen la aplicación, siendo en base a estos que se realizan los cambios necesarios y a partir de estos se regenera el código que constituye el sistema operacional. Para esto se deben incorporar actividades, roles y entregables específicos para el enfoque, así como una herramienta de desarrollo que lo soporte.

4.1 Proceso de desarrollo basado en RUP

El principal proceso de desarrollo base con que cuenta la Organización, en este caso el Grupo de Ingeniería de Software (Gris) en el curso "Proyecto de Ingeniería de Software", es una adaptación del Rational Unified Process (RUP) que ha ido evolucionando desde su primera versión que fuera definida por estudiantes en el año 2000 como proyecto de grado [9]. A partir del año 2004 se compone de un esqueleto de actividades que es común a todos los tipos de desarrollo que se emprendan, y dos extensiones principales para realizar desarrollos específicos en OO (Orientación a Objetos) y Genexus [10], herramienta de cuarta generación para el desarrollo de aplicaciones orientadas a bases de datos. En este contexto la metodología de desarrollo para aplicaciones con enfoque MDA se incorpora al esqueleto base más la extensión OO, agregando actividades, roles y entregables específicos, así como una herramienta adecuada para el desarrollo con dicho enfoque.

El proceso base tiene como el RUP dos dimensiones, el tiempo y las disciplinas. En la dimensión del tiempo se definen cuatro fases: Inicial, Elaboración, Construcción y Transición, en las tres primeras fases se definen dos iteraciones de dos semanas cada una, y en la última fase una iteración de dos semanas, que completan junto con la semana de preparación previa y la semana de evaluación final, las dieciséis semanas

con que cuenta el curso. Se cuenta además con una agenda definida que indica para cada semana que actividades se deben realizar y que entregables se deben obtener. En la dimensión de las Disciplinas se definen las disciplinas tradicionales del desarrollo de software: requerimientos, diseño, implementación y verificación, más las disciplinas de soporte: gestión del proyecto, de la calidad, de la configuración e implantación. En cada disciplina se definen actividades, roles encargados y participantes de las actividades, y entregables de entrada y salida de dichas actividades. Una descripción completa del proceso base incluyendo su concepción y evolución puede consultarse en [5].

Roles, Actividades y Entregables en MDA

En la definición del Proceso Extensión MDA, no se hicieron cambios en los roles ni a las combinaciones de roles definidos en el proceso base, siendo los participantes más importantes el Arquitecto, los analistas y los Especialistas Técnicos. Se vió que era necesario contar con algún asistente de Arquitecto, ya que éste se veía sobrecargado de trabajo. También se detecto la necesidad de que uno de los Especialistas Técnicos se dedicara solamente a la herramienta MDA, ya que el aprendizaje de la misma, requiere dedicación y pruebas. Como hay que modelar de forma tal que la herramienta MDA pueda traducir el modelo en código, es necesario aprender para que sirve cada estereotipo y tagged value por tecnología. Los perfiles UML son los que permiten agregar esta información adicional al modelo y que la herramienta interprete como generar el código, dependiendo de las marcas que se le asignen a los diferentes objetos del modelo. Esta es la tarea principal del Especialista MDA, quien luego transmite su conocimiento al resto de los integrantes del equipo.

La aplicación del enfoque MDA comienza desde la captura de requerimientos en el CIM, el diseño del PIM en la herramienta seleccionada, y la generación del PSM y/o código basado en las tecnologías a utilizar. En la disciplina de Requerimientos no se agregan actividades sino que se conceptualiza el enfoque agrupando actividades que ya existen. En la disciplina de Diseño se agregan las actividades más importantes del enfoque que tienen que ver con el modelado del PIM y el marcado del mismo para poder generar el código correspondiente, tarea que se realiza en la disciplina de Implementación. Además, para cada actividad se definen los roles responsables, participantes y sus entregables de entrada y salida.

Disciplina Requerimientos

En esta disciplina participan principalmente los roles de Analista y Arquitecto, quienes realizan principalmente el relevamiento de requerimientos a partir de reuniones con el cliente. Para el enfoque MDA no se realizan agregados de actividades sino que se define la conceptualización del CIM. Este modelo, generalmente se hace también en el proceso base, pero consta de tres partes: la especificación de requerimientos, el modelo de casos de uso y el modelo de dominio.

Especificar el CIM (R11): tiene como objetivo la conceptualización del CIM y consta de la reunión de los tres documentos como el modelo mencionado. El rol responsable de este entregable es el analista.

Disciplina Diseño

En la disciplina de Diseño, participan los roles de Analista, Arquitecto y los Especialistas técnicos, donde los primeros aportan su conocimiento de los requerimientos del cliente, el Arquitecto traduce los requerimientos en el diseño de la solución al problema planteado, mientras los últimos aportan el conocimiento técnico de las herramientas. El enfoque MDA agrega dos nuevas actividades que se realizan además de las ya definidas en el proceso base, que tienen como rol principal al arquitecto, ayudado por analistas y especialistas técnicos.

Especificar el PIM (D6): consiste en crear un modelo del sistema en alto nivel. También se compone de salidas de actividades definidas en el proceso base, como la Descripción de la Arquitectura, el modelo de diseño, diagramas de Subsistemas, Clases, Colaboración, Secuencia, y el modelo de datos. El diagrama de clases se hace en UML, que luego es marcado con estereotipos y tagged values para conformar el PIM marcado que es el resultado de la siguiente actividad.

Definir Marcas para el PIM (D7): como ya se menciono, se agrega información adicional al modelo en UML, para que este pueda ser transformado automáticamente, dependiendo de la tecnología de desarrollo a utilizar. Junto con el PIM marcado se entrega una planilla de marcas, que tiene como objetivo facilitar la tarea de estereotipado, y consta de un cuadro que indica a que clase le corresponde que estereotipo y quien es el encargado de dicha clase.

Disciplina Implementación

En esta disciplina se encuentran definidas actividades para realizar la implementación del diseño definido, y los roles participantes son Especialistas Técnicos e implementadores. Para el enfoque MDA se modifican actividades y se agregan nuevas.

Investigar la herramienta de desarrollo (I8): esta actividad es del proceso base y se le agrega el estudio de la o las herramientas que se utilizarán para MDA.

Crear las clases y subsistemas (I10, I11): son actividades del proceso base que se ven modificadas pues el trabajo de generación de código manual se espera sea mucho menor, ya que la mayoría del código es generado automáticamente por la herramienta MDA.

Especificar el PSM (I13): consiste en transformar el PIM marcado obtenido anteriormente, para obtener un modelo especifico de la plataforma a utilizar, en caso que se utilice más de una plataforma será uno por cada plataforma elegida.

Definir características del proyecto (I14): el objetivo de esta actividad es configurar el ambiente de desarrollo, con la herramienta MDA a utilizar en el proyecto. Esta información se almacena en una planilla de forma que la misma sea un referente para el equipo a la hora de trabajar con la herramienta MDA.

Selección de herramientas MDA

Para poder extender el proceso con el enfoque MDA, además de definir las actividades y entregables, se debía proporcionar la herramienta MDA. Para ello se investigaron algunas de las herramientas existentes en el mercado, buscando una que además de cumplir con el enfoque, fuera factible su uso en el curso "Proyecto de Ingeniería de Software" que plantea algunas restricciones como el uso de Eclipse [11] como IDE para el desarrollo. La selección consto de tres etapas: investigación de herramientas existentes, preselección de herramientas, y selección de la herramienta MDA a utilizar en el curso. Para definir una metodología de evaluación de herramientas, se utilizo como referencia el estudio en [12].

Investigación de herramientas MDA

Como salida de la investigación de herramientas se obtuvo una planilla de herramientas disponibles y sus características según la evaluación realizada, que constituyó la base para la siguiente etapa. Algunas de las características que se consideraron mas importantes para esta etapa se muestran en la Tabla 1:

Propiedad	Descripción
Licenciamiento	Free o si es comercial período de uso
Documentación	Existencia y calidad de la documentación
Plug-in Eclipse	Si es plug-in para Eclipse y de que versión
Reqs. de CPU disco, memoria	Requerimientos para instalación y ejecución
Generación de código	Tecnologías para las que genera código
Herramienta de modelado	Si incluye o no herramienta de modelado
Estándar UML	Versión UML 1.4 y/o 2.0
Estándar XMI	Versión XMI 1.x y/o 2.x

Tabla 1: criterios para evaluación de herramientas MDA

Las herramientas MDA evaluadas fueron: ArcStyler 5.5 Architect Edition, OptimalJ Professional Edition 4.1, IBM Rational Software Architect 6.0, AndroMDA 3.2, AndroIDE 0.0.5, Acceleo 1.0.1, TaylorMDA 0.0.2 y OpenMDX 1.12.1. La evaluación de características para algunas de las herramientas mencionadas se muestra en la tabla 2:

Propiedad	IBM RSA 6.0	AndroMDA 3.2
Licenciamiento	Free por 30 días.	Open Source

Documentación	Excelente+ videos explicativos	Buena
Plug-in Eclipse	Si	No
Reqs. de CPU disco, memoria	6 Gb disco, y > 512 Mb RAM	400 Mb de disco aprox.
Generación de código	C#, Java y EJB	Java, .NET y generar nuevos cartuchos
Herramienta de modelado	Trae incorporada en la herramienta MDA	No tiene, se debe combinar con alguna
Estándar UML	UML 2.0	UML 1.4 y 2.0
Estándar XMI	XMI 2.0	XMI 1.2

Tabla 2: evaluación de propiedades para herramientas MDA

Preselección de herramientas MDA

De la primera evaluación, se eligieron dos herramientas: una herramienta comercial Rational Software Architect de IBM [13] para la cual se cuenta con licencia universitaria, y una herramienta Open Source AndroMDA[14], que debía ser usada conjuntamente con una herramienta de modelado compatible ya que como la mayoría de las herramientas MDA que son open source, no es completa. Para poder utilizarla fue necesario estudiar que herramientas de modelado en UML hay disponibles, y cuales eran compatibles y factibles de usar según los requerimientos del curso. En la tabla 3 se muestra la evaluación de las principales características para algunas de las herramientas de modelado compatibles con AndroMDA:

Propiedad	MagicDraw 9.5	Poseidon 3.2.1	Eclipse UML
Licenciamiento	CE-free PE-licencia	CE-free PE-licencia	20 dias, licencia academica
Plataforma	Windows /Linux	Windows / Linux	Windows / Linux
Documentación	Buena	Buena	Buena
Plug-in Eclipse	No	No	Si
Estandar UML	1.4	1.4	1.4 y 2.0
Estandar XMI	1.2	1.2	2.0
Exportar XMI	Si	Si	No

Importar XMI	Si	Si	No

Tabla 3: evaluación de propiedades para herramientas de modelado

En un principio, se selecciono una herramienta de modelado que es plug-in de Eclipse pero no es gratuita: EclipseUML Studio Edition de Omondo[15]. Se obtuvo una licencia universitaria que permitía utilizarla en el curso. Esta herramienta ofrece la posibilidad de importar y exportar modelos, característica que otras herramientas no permitían o solo permitían exportar un archivo de imagen con el modelo, en lugar del modelo en si. Sin embargo, en la página de AndroMDA se indica que la herramienta es compatible pero puede tener problemas con la versión de XMI, por lo que se selecciono la segunda en la lista de recomendadas AndroMDA: Poseidon versión 3.2.1 [16] que posee una versión free, pero no es plugin de eclipse. Actualmente AndroMDA es compatible con versiones mas nuevas de las herramientas de modelado.

Selección de herramientas MDA

Finalmente se debió decidir entre las dos herramientas preseleccionadas: Rational Software Architect y AndroMDA con Poseidón para el modelado. Teniendo en cuenta que la utilización de RSA implicaba realizar todo el desarrollo, incluyendo el modelado, en la misma herramienta, y dado que consume muchos recursos y las maquinas con que cuenta la facultad, no eran capaces de soportarla, se optó por utilizar la segunda opción, AndroMDA, que solo requiere la instalación de maven y un plugin, utilizando una consola para generar el código, que puede ser editado utilizando Eclipse. Además es desarrollada por la comunidad, por lo que creemos que tiene valor su utilización para generar aportes a dicho desarrollo.
Sin embargo, durante el desarrollo de la Fase Inicial se encontraron problemas con la herramienta de modelado elegida, porque no era posible representar todo lo necesario en los modelos y por algunas incompatibilidades y falta de soporte en AndroMDA. Al consultar en el foro de AndroMDA, la sugerencia fue utilizar Magic Draw [17], pues es la usada por sus desarrolladores, por lo que es posible obtener soporte rápidamente. Esta herramienta se había descartado en un principio porque la licencia académica parecía no ser gratis y la edición CE tenía limitaciones con respecto al número de diagramas que se podían crear. Se realizaron entonces consultas con la empresa a fin de obtener una licencia académica, que fue concedida completamente gratis. Debido a ello, se decidió cambiar a dicha herramienta de modelado. La suite completa de herramientas seleccionadas para el proceso MDA definido se muestra en la figura 2:

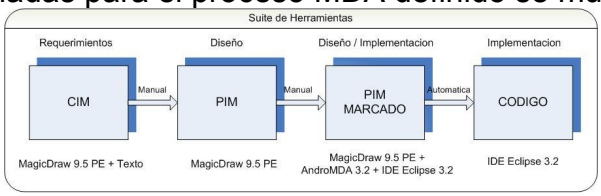

Figura 2. Suite de herramientas en el proceso MDA

Aplicación de la Extensión MDA

Para realizar la prueba de la Extensión MDA definida en el curso Proyecto de Ingeniería de Software correspondiente al segundo semestre del 2006, se eligió un proyecto que tuviera características que permitieran probar las capacidades del enfoque y de las herramientas seleccionadas. El Proyecto LIMS (Laboratory Information Management System) llevado adelante por el Instituto de Computación y el

Instituto Pasteur de Montevideo IPMONT-FING [18] fue elegido debido principalmente a que planteaba varios desafíos en las tecnologías a utilizar y la reutilización de código existente ya desarrollado en otros proyectos.

Descripción del proyecto

Los LIMS son Sistemas para gestión de información en laboratorios, registran la información generada en los procesos y experimentos, y permiten manipular la información integrada facilitando: acceso compartido a resultados, según permisos de acceso que se otorguen, e implementación de mecanismos de colaboración entre investigadores y/o grupos. Si bien existen productos comerciales que proveen varias de estas funcionalidades, resultan poco adaptables a requerimientos específicos, por lo que surgen productos con fuerte componente académica por la necesidad de contar con herramientas especializadas.

El proyecto plantea la construcción de un LIMS según los requerimientos de IPMONT, con utilización de tecnologías abiertas como J2EE [19], servidores con Linux[20], servidor de aplicaciones Jboss[21] integrado con Jboss Portal[22] y JBPM[23], base de datos Postgress[24], y basado en estándares y otros proyectos como el EBI-PIMS[25]. El diseño de la aplicación tendrá fuerte orientación a servicios (Service Oriented Architecture SOA) con interfaces normalizadas, permitiendo la definición de módulos que sean integrados en forma incremental, y acceso Web que permita aumentar opciones de acceso y usabilidad y procesamiento distribuido, así como colaboración multi-institucional.

En cuanto al proyecto específico definido para el curso Proyecto de Ingeniería de Software, los requerimientos incluyen funcionalidades de workflow de experimentos, movilidad de muestras en cada laboratorio y entre laboratorios, y gestión de proyectos que incluyen la realización de varios experimentos. Para el manejo de los elementos del dominio se reutilizará código de la herramienta desarrollada en el marco del EBI-PIMS, reutilizando el conocimiento biológico existente en dicho proyecto, como ser definición de samples y targets de experimentos.

Prueba de la Extensión MDA

Para realizar la prueba de la Extensión MDA se asignó dicho proyecto a dos grupos de estudiantes identificados como grupos 1 y 2, como forma de tener dos ocurrencias distintas del mismo proceso de desarrollo con el mismo proyecto, y poder evaluar los resultados obtenidos a la luz también de las comparaciones posibles entre los dos grupos y el desarrollo del mismo según las características de cada uno. Como primer tarea se presentó el proceso MDA definido, para lo cual se les brindó a los grupos la base conceptual del enfoque y de las actividades, entregables, roles y responsabilidades definidas. A medida que los grupos fueron avanzando en los requerimientos y modelado en la herramienta seleccionada, se realizó soporte de la misma en cuanto a aspectos relacionados con la realización del PIM y del PIM marcado, entrada para la generación con AndroMDA. Como se mencionó, al final de la Fase Inicial fue necesario que cambiar la herramienta de modelado ya que se detectaron algunas incompatibilidades y no era posible obtener soporte desde AndroMDA para la misma. Esta decisión creemos que fue adecuada para el avance de

los proyectos, ya que a partir del cambio a MagicDraw pudieron generar el código en AndroMDA según las necesidades del proyecto.

La herramienta AndroMDA genera el código basado en el cartucho elegido, el cual provee los estereotipos necesarios para marcar las distintas clases e interpretar lo que se espera obtener. En la figura 3 se presenta un ejemplo de PIM marcado realizado por uno de los grupos que está siguiendo el proceso MDA:

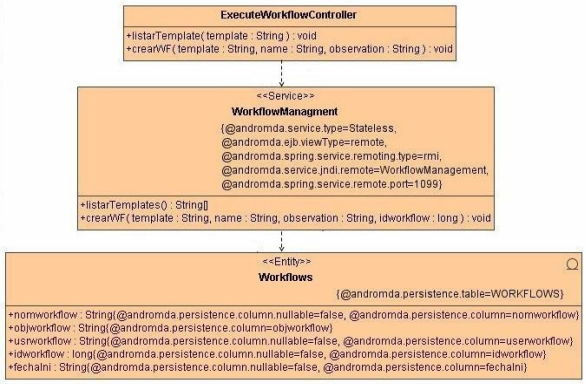

Figura 3. Ejemplo de PIM marcado en Magic Draw

Entre los cartuchos utilizados por los grupos para el proyecto se encuentran el JSF[26] para la presentación, Spring[27] y J2EE para la lógica, JBPM para la generación de workflows, y Postgress para la persistencia. La generación provee la aplicación de patrones de diseño como ser Data Access Object(DAO) con Hibernate[28] para el acceso a datos, obteniendose en este caso una generación del 100% del código incluyendo las clases para el mapeo entre paradigmas OO y relacional, y los scripts para la generación del esquema de la BD. Para otros requerimientos como ser la generación de portlets para el Jboss Portal no hay aún cartucho que permita realizarla desde la herramienta, por lo que este aspecto se tuvo que implementar sin generación.

Seguimiento de la prueba Extensión MDA

Para el seguimiento de la prueba de la Extensión MDA definida, según lo establecido en el proceso base, se cuenta con dos actividades principales: la revisión semanal con el Director de Proyecto que en este caso es quién también dirige el proyecto de grado de los estudiantes que definieron la Extensión MDA, y Auditorías a los grupos por parte de los docentes del curso, en este caso, conjuntamente con los estudiantes que definieron el proceso. Una descripción detallada de estas actividades puede encontrarse en [5].

Se realizaron hasta el momento dos Auditorías con los grupos, una al finalizar la Fase Inicial y otra al finalizar la Fase de Elaboración. Ambos grupos se atrasaron en el cumplimiento de los objetivos de la Fase Inicial, debido principalmente a aspectos relacionados con las tecnologías a utilizar, incluida la primer herramienta de modelado definida para MDA. En Fase de Elaboración se detectaron problemas en cuanto al procedimiento seguido para el manejo del modelo correspondiente al PIM marcado, ya que un grupo decidió que varias personas realizaran esta actividad a la vez, para lo cual cada uno se debía llevar una copia del modelo exportado en formato XMI, definiendo sobre que package trabajaría cada uno. Al momento de la integración se encontraron dificultades para volver a obtener un único modelo, lo que demandó más esfuerzo de integración del esperado.

A esta altura del proyecto los grupos han validado con el cliente el ejecutable de la Línea Base de la Arquitectura obtenido en la Fase de Elaboración, y que implementa los Casos de Uso relevantes a la Arquitectura. A pesar de las dificultades encontradas, ambos grupos han podido generar un alto porcentaje del código requerido para la aplicación en desarrollo, luego del cambio en la herramienta de modelado a Magic Draw, que se integra perfectamente con la herramienta de generación AndroMDA. El cliente ha validado el cumplimiento de los requerimientos básicos para la aplicación por parte de ambos grupos y las soluciones encontradas. Actualmente los proyectos están en Fase de Construcción, incorporando el resto de las funcionalidades incluidas en el Alcance para obtener el ejecutable final del proyecto.

Conclusiones y trabajo futuro

La Extensión MDA propuesta incorpora a las actividades, entregables y roles establecidos por el proceso base adaptación del RUP, la conceptualización del enfoque que se ve reflejada en las modificaciones y agregados definidos. Los aspectos clave del desarrollo basado en modelos se definen en las disciplinas de Requerimientos donde se obtiene el CIM, en la de diseño donde se realizan el PIM y el PIM marcado y en la de implementación donde se genera el PSM y/o el código a partir de los modelos.

Las actividades y entregables definidos para el proceso MDA son independientes de la o las herramientas que se utilicen para ponerlo en práctica, y está basado en la definición del enfoque provista por el estándar de la OMG. Esto permite que pueda ser aplicado utilizando distintas herramientas que cumplan los requerimientos establecidos, lo que abre las puertas para la evaluación de herramientas disponibles. La curva de aprendizaje del enfoque y de la herramienta puede resultar alta al principio del proyecto, pero una vez que se maduran los conceptos y se comprende el funcionamiento de la herramienta, el código generado permite un rápido avance en el desarrollo. Se realizarán evaluaciones de la calidad del código generado, por ejemplo la aplicación de patrones de diseño.

Para el próximo año se piensa conjuntar la Extensión MDA con la Extensión SOA realizada para desarrollos con enfoque Service Oriented Architecture (SOA) en el año 2005 como parte de la tesis de maestría de un docente del Gris según [29], [30] y [31].

REFERENCIAS

[1] Grupo de Ingeniería de Software (Gris), Instituto de Computación, Facultad de Ingeniería, Universidad de la República, <http://www.fing.edu.uy/inco/grupos/gris/> [Consulta: diciembre de 2006]

[2] Proyecto de Ingeniería de Software, Procesos, Instituto de Computación, Facultad de Ingeniería, Universidad de la República,<http://www.fing.edu.uy/inco/cursos/ingsoft/pis/>[Consulta: diciembre de 2006]

[3] IBM Rational Unified Process. <http://www-130.ibm.com/developerworks/rational/products/rup/> [Consulta: diciembre de 2006]

[4] Beck, K. Extreme Programming Explained: Embrace Change, Addison Wesley Professional, 1999, ISBN 201-61641-6

[5] Delgado, A. Pérez, B. "Modelo de Desarrollo de Software OO – Experimentación en un curso de Ingeniería de Software", en V Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC'06), Puebla, México, Febrero de 2006, ISBN 970-94770-0-5

[6] Extensión MDA <http://www.fing.edu.uy/~pgmdapis> [Consulta: diciembre de 2006]

[7] Object Management Group (OMG), Model Driven Architecture (MDA), <http://www.omg.org/mda/> [Consulta: diciembre de 2006]

[8] Kleppe A., Warmer J., Bast W. MDA Explained, The Model Driven Architecture: practice and promise, Addison-Wesley Pearson Education, 2003. ISBN 0-321-19442-X

[9] Delgado, A. Pérez, B. Modelado del proceso de software - Informe final Proyecto de Taller V, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, año 2000

[10] Visión General, ARTech, Febrero 2003, <http://www.genexus.com/DOCUM/GeneXus_VG.pdf> [Consulta: diciembre de 2006]

[11] Eclipse, <http://www.eclipse.org> [Consulta: diciembre de 2006]

[12] King's College London, An evaluation of Compuware OptimalJ Professional Edition as an MDA Tool, 2003, <http://www.compuware.com/dl/kings_mda.pdf > [Consulta: diciembre de 2006]

[13] RationalSoftwareArchitect,<http://www-128.ibm.com/developerworks/rational/products/rsa/>[Consulta: diciembre de 2006]

[14] AndroMDA, <http://www.andromda.org> [Consulta: diciembre de 2006]

[15] EclipseUML Studio Edition, <http://www.omondo.com> [Consulta: diciembre de 2006]

[16] Poseidón, <http://www.gentleware.com> [Consulta: diciembre de 2006]

[17] Magic Draw, <http://www.magicdraw.com> [Consulta: diciembre de 2006]

[18] Proyecto LIMS, IPMONT-FING <http://www.fing.edu.uy/inco/pm/Convenios/LIMS2006> [Consulta: diciembre de 2006]

[19] J2EE, <http://java.sun.com/j2ee/> [Consulta: diciembre de 2006]

[20] Linux, <http://www.linux.org/> [Consulta: diciembre de 2006]

[21] JBoss AS, <http://www.jboss.com/products/jbossas> [Consulta: diciembre de 2006]

[22] JBoss Portal, <http://www.jboss.com/products/> [Consulta: diciembre de 2006]

[23] JBPM, <http://www.jboss.com/products/jbpm> [Consulta: diciembre de 2006]

[24] PostgreSQL, <http://www.postgresql.org> [Consulta: diciembre de 2006]

[25] Proyecto EBI-PIMS <http://www.ebi.ac.uk/> , <http://www.pims-lims.org/svn/pims/frontpage/index.html> [Consulta: diciembre de 2006]

[26] JSF, <http://java.sun.com/javaee/javaserverfaces/> [Consulta: diciembre de 2006]

[27] Spring, <http://www.springframework.org/> [Consulta: diciembre de 2006]

[28] Hibernate, <http://www.hibernate.org/> [Consulta: diciembre de 2006]

[29] Delgado A. Metodología de desarrollo de aplicaciones con enfoque SOA (Service Oriented Architecture) Agosto 2005, <http://www.fing.edu.uy/~adelgado/ExtensionSOA/index.htm> [Consulta: diciembre de 2006]

[30] Delgado A., González L., Piedrabuena F. "Desarrollo de aplicaciones con enfoque SOA (Service Oriented Architecture)", en V Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC'06), Puebla, México, Febrero de 2006, ISBN 970-94770-0-5

[31] Delgado, A., Metodología de desarrollo para aplicaciones Service Oriented Architecture (SOA), en XXXII Confererencia Latinoamericana de Informática (CLEI'06), sesión 4, número artículo 265, Santiago de Chile, Chile.

Agradecimientos

El presente trabajo ha sido desarrollado en el marco del proyecto COMPETISOFT (Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria de Software de Iberoamérica) del programa CYTED (Ciencia y Tecnología para el Desarrollo).