PROCESOS Y HERRAMIENTAS PARA DESARROLLO CONDUCIDO POR MODELOS

Servetto, A. (aserve@gmail.com); **Grossi M. D.** (mdg7501@yahoo.com.ar); **Jiménez Rey, M. E.** (ejimenezrey@yahoo.com) / **Lab. de SSOO y BBDD**
López, G.; **Jeder, I.**; **Echeverría, A.** ({glopez, jeder, aechevi}@fi.uba.ar) / **Lab. de Informática de Gestión**
Departamento de Computación / Facultad de Ingeniería / Universidad de Buenos Aires

RESUMEN

Se trabaja con esta tecnología de desarrollo de software, basada en el uso de modelos o abstracciones más cercanos a conceptos y procedimientos de un dominio de información que a su representación algorítmica, para generar código a partir de ellos.

El objetivo principal es encontrar primitivas de abstracción con representación gráfica que permitan integrar interfaces y comportamiento de un sistema informático de manera de maximizar la generación de código.

También se trabaja en la especificación de una propuesta para la estandarización del proceso automatizable de desarrollo y en la determinación de métricas para evaluar la complejidad de sistemas a desarrollar que permitan una planificación efectiva.

La iniciativa se limita a los sistemas de gestión y a sus patrones característicos de estructuras y procesamiento de datos.

Palabras clave: *desarrollo conducido por modelos, MDD, procesos MDD, arquitectura conducida por modelos, MDA, modelos independientes de plataformas, MDI, modelos específicos de plataforma, PSM.*

CONTEXTO

La línea de investigación corresponde a los Laboratorios de Bases de Datos y Sistemas Operativos y de Informática de Gestión del Departamento de Computación de la Facultad de Ingeniería de la Universidad de Buenos Aires.

1. INTRODUCCION

El Desarrollo Conducido por Modelos o MDD (Model-Driven Development) es una tecnología de desarrollo de software donde los principales componentes son los modelos, a partir de los cuales se genera el código y otros componentes. El MDD usa los modelos para capturar los requerimientos y automatizar parcial o totalmente la implementación.

Los propósitos del MDD son:

- Elevar notablemente el nivel de abstracción para que la programación no sea el centro de atención.
- Acelerar el desarrollo de software y hacerlo más eficiente, mediante el uso de modelos para generar el código.

El MDD también separa la lógica del negocio de las aplicaciones de plataformas de desarrollo.

Un modelo es una descripción particular de un sistema desde una particular perspectiva, omitiendo detalles irrelevantes para que las características de interés se vean con más claridad. Utilizando modelos podemos centrarnos en el diseño lógico de la aplicación, y permite liberarnos de los detalles de la implementación.

El esfuerzo que se invierte en el modelado tiene una continuidad durante el desarrollo, ya que el objetivo es que dirijan de forma automatizada el desarrollo del código durante todas las fases e iteraciones del proyecto y que los modelos no sean meramente parte de la documentación.

El MDD no es exactamente un estándar, sino la visión y la forma de llevar al mundo del desarrollo de software la idea de obtener código generado automáticamente, a través de transformación de modelos y generación automática de código. Tiene como ventaja el pragmatismo, que es una realidad y su evolución. Su desventaja actual es la falta de estándares.

Un desarrollo con esta tecnología distingue al menos las siguientes etapas:

- Construir un modelo independiente de plataforma o PIM (Plataform Independent Model) en un alto nivel de abstracción, independiente de cualquier tecnología específica.
- Transformar al PIM en uno o más modelos dependientes de una plataforma específica, denominados modelos específicos de plataforma o PSM (Platform Specific Model), por ejemplo relacional, J2EE, .NET
- Transformar los PSM a código.

La unión de una serie de progresos en el campo de las tecnologías de la información ha hecho posible la existencia de las tecnologías de MDD/MDA (Model Driven Arquitecture). Los principales avances son los siguientes:

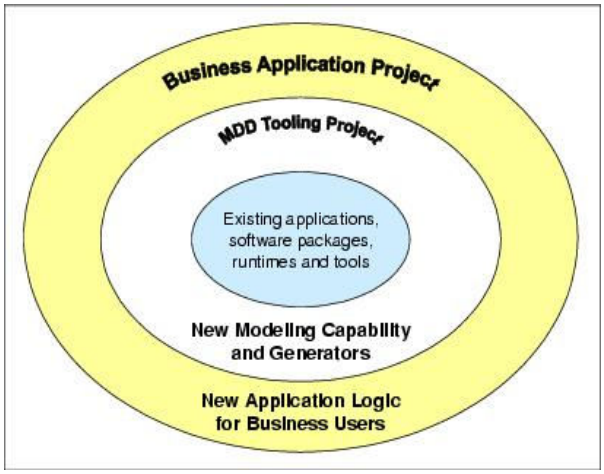
- Lenguajes Orientados a Objetos: se centran en las distintas clases de elementos que aparecen en la aplicación, permiten la especialización de clases y unifican datos y procesos; todo esto facilita la generación automática de código, frente a los módulos monolíticos de otros tipos de programación.
- UML (Lenguaje de Modelado Unificado): permite representar un sistema de información bajo distintos puntos de vista y a gran nivel de detalle. Tienen una especial importancia las posibilidades de extensión de su semántica.
- XMI (XML Metadata Interchange): formato estándar de XML para almacenar modelos de sistemas de información expresados en UML.
- QVT (Query View Transformation): lenguaje estándar creado por el OMG Object Management Group), mediante el cual se pueden hacer consultas y transformaciones a modelos de sistemas expresados en UML.

Los procesos de desarrollo con esta tecnología implican:

- Utilización de más tiempo en análisis y diseño.
- Menor tiempo de codificación.
- Menos defectos por perdida de información en la comunicación, y mayor calidad.
- Integración continúa.
- Ciclos de desarrollo más ágiles.

El MDD tiene un efecto profundo sobre la forma en que se construye software de aplicaciones comerciales. Recoge la experiencia y las decisiones de la gente más técnica y la pone a disposición del resto del equipo a través de herramientas personalizadas para las necesidades de su proyecto. El costo de las fases de desarrollo y de pruebas del software se reduce considerablemente ya que mucho del trabajo de bajo nivel de codificación queda automatizado. El número de errores se reduce y hay un aumento en la coherencia con que se lleva a cabo el trabajo.

Esto suena ideal, pero ¿cuál es la trampa? Básicamente, un Líder de Proyecto debe controlar un proyecto dentro de otro. El proyecto interior es el desarrollo de herramientas MDD (que se utilicen para la construcción de la aplicación empresarial en el proyecto externo) que el equipo de desarrollo puede utilizar. Con estas circunstancias se debe organizar y planificar con cuidado, especialmente al inicio del proyecto.



La iniciativa de MDD más conocida es la del OMG (Object Management Group): MDA (Model Driven Architecture).

La MDA data del año 2000, cuando el OMG publicó un white-paper titulado "Model Driven Architecture", en el que describía la visión del desarrollo de software a través de modelos de objetos relacionados entre sí, para la generación de sistemas completos. Es un framework para el desarrollo de software en el que el proceso de desarrollo de software está dirigido por la actividad de modelar el software.

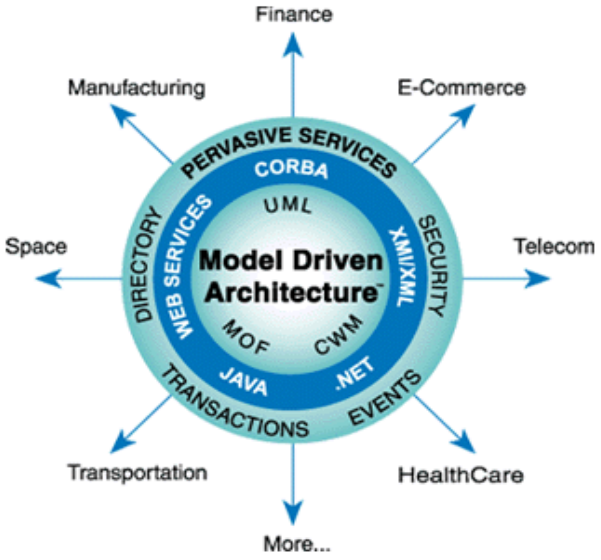
Uno de los principales objetivos de la MDA es separar el diseño de la arquitectura y de las tecnologías de construcción, facilitando que el diseño y la arquitectura puedan ser alterados independientemente. El diseño alberga los requerimientos funcionales (casos de uso) mientras que la arquitectura proporciona la infraestructura a través de la cual se hacen efectivos requerimientos no funcionales como la escalabilidad, fiabilidad o rendimiento.

La MDA se asegura de que el modelo independiente de la plataforma (PIM), sobreviva a los cambios que se produzcan en las tecnologías de fabricación y en las arquitecturas software.

De particular importancia en la MDA es la noción de transformación de modelos. Uno de los estándares definidos para la transformación de modelos se denomina QVT (Query View Transformation).

La MDA utiliza los estándares abiertos de modelado establecidos por la OMG:

- Unified Modeling Language (UML).
- Meta-Object Facility (MOF).
- Common Warehouse
- Metamodel (CWM).



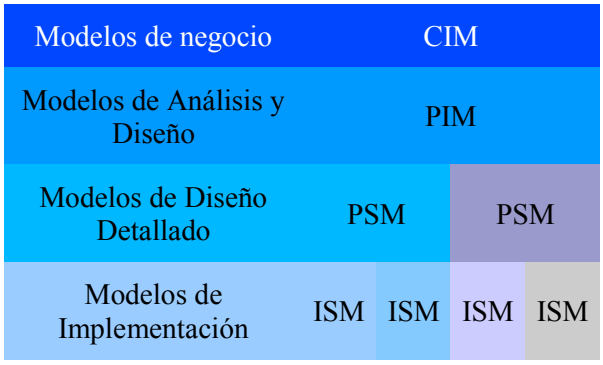
El OMG define cuatro principios que sustentan el enfoque de MDA:

1. Los modelos expresados en una notación bien definida son la pieza clave del sistema de comprensión de soluciones de escala empresarial.
2. La creación de sistemas pueden ser organizada en torno a un conjunto de modelos mediante la imposición de una serie de las transformaciones entre modelos, organizados en un marco arquitectónico de las capas.
3. La descripción de modelos se basa en un conjunto de metamodelos que facilitan la integración y la transformación entre modelos, y es la base para la automatización a través de herramientas.
4. La aceptación y la amplia adopción de este enfoque exige a la industria normas para brindar transparencia a los consumidores, y fomentar la competencia entre los proveedores.

Para apoyar este enfoque el OMG ha definido un conjunto específico de capas y transformaciones que proporcionan un marco

conceptual y un vocabulario para la MDA. En particular, el OMG identifica cuatro capas, corespondientes a las fases tradicionales del desarrollo de software:

1. Modelo Independiente de Computación (CIM)
2. Modelo Independiente de Plataforma (PIM)
3. Modelo Específico de Plataforma (PSM)
4. Modelo Específico de Implementación (ISM)



Un aspecto clave del enfoque de la MDA es reconocer que las transformaciones se pueden aplicar a descripciones abstractas de algún aspecto de un sistema para añadir más detalles a la descripción, refinar la descripción, o para convertir entre representaciones. Tres ideas son importantes:

1. Distinguir los diferentes tipos de modelos permite pensar en el software y en el proceso de desarrollo como una serie de mejoras entre las representaciones de modelos diferentes. Estos modelos y sus mejoras son una parte crítica del desarrollo de la metodología para las situaciones que incluyen mejoras entre los modelos que representan diferentes aspectos del sistema, agregando más detalles a un modelo, o la conversión de entre los diferentes tipos de modelos.
2. Una forma de considerar los modelos es clasificarlos en términos de cómo representan aspectos de las plataformas como objetivo. En todo el software y el proceso de desarrollo existen importantes limitaciones que implica la elección de los lenguajes, hardware, la topología de red, protocolos de comunicación e infraestructura, etc. Cada una de estas cosas puede ser considerada parte de la "plataforma". La MDA nos ayuda a centrarnos en la solución que se diseñó separada de los detalles de la “plataforma".
3. La noción de lo que es una "plataforma" es bastante compleja, y altamente dependiente del contexto. Por ejemplo, en algunas situaciones puede ser la plataforma del sistema operativo y los servicios públicos asociados, en algunas situaciones puede ser una infraestructura de tecnología representada por un modelo de programación bien definido, como J2EE o .NET, en otras situaciones, es un caso particular de una topología de hardware. Independientemente de lo que se considere "plataforma", es importante pensar más en términos de modelos en los diferentes niveles de abstracción utilizados para diferentes fines, en lugar de distraerse también por definir qué es una "plataforma".

Un amplio conjunto de herramientas con soporte para MDA están siendo desarrolladas por los principales fabricantes y proyectos Open Source. Estas herramientas permiten comúnmente la especificación rudimentaria de arquitecturas. Algunos otros frameworks de MDA son:

- OpenMDX: www.openmdx.org, es una plataforma MDA Open Source basada en los estándares MDA de la OMG.
- Eclipse Modeling Project: www.eclipse.org/modeling. Está enfocado en la evolución y promoción de las tecnologías de desarrollo dirigido por modelos en la comunidad eclipse.

- Acceleo: http://acceleo.org/pages/home/en. Es una herramienta para la construcción de generadores de código basada en eclipse.
- OpenArchitectureWare (oAW): www.openarchitectureware.org. Es un framework de generación MDA/MDD modular basada en eclipse. Soporta EMF, y otros modelos en UML2, XML o JavaBeans.
- Rational Architect: Es una herramienta de IBM para Model Driven Development.
- Enterprise Architect: http://www.sparxsystems.com/, es una herramienta UML para desarrollo y modelado de software.
- AndroMDA: http://www.andromda.org motor de transformación de modelos en código fuente y/o archivos de configuración extensible

2. LINEAS DE INVESTIGACION y DESARROLLO

Propagación automática de cambios: la generación de código partir de modelos no es totalmente automática. Se trabaja en la determinación de transformaciones que permitan lograr que una modificación en un modelo se propague al código ya generado manteniendo la integridad.

Integración de modelos: distintos modelos de un mismo sistema suponen vistas distintas pero independientes y muy difíciles de integrar. Se trabaja en la determinación de convenciones y mecanismos para integrar modelos de manera que permitan maximizar la cantidad de código generado.

Ambientes de modelado: existen muchos editores de modelos pero pocos permiten compilarlos, optimizarlos y/o transformarlos consistentemente, o lo hacen en forma parcial. Se trabaja en la determinación de reglas de validación, optimización y transformación de modelos así como en la definición de métricas para calcular complejidad que puedan integrarse a un ambiente.

3. RESULTADOS OBTENIDOS/ESPERADOS

Se ha efectuado un relevamiento, evaluación y comparación de herramientas de MDD. Se pretende construir un ambiente que integre la edición, validación e integración de modelos independientes de plataforma, la estandarización de esos modelos, la compilación o transformación de los modelos integrados a modelos dependientes de plataforma y a código ejecutable.

4. FORMACION DE RECURSOS HUMANOS

Los autores son tutores de trabajos de investigación y/o desarrollo en materias de las carreras de Ingeniería en Informática (Tesis y Trabajo Profesional) y de Licenciatura en Análisis de Sistemas (Aplicaciones Informáticas). Este año se aprobó un trabajo de la materia Aplicaciones Informáticas con un desarrollo usando la herramienta Acceleo y actualmente dos alumnos están desarrollando una tesina sobre el tema para la misma materia.

5. BIBLIOGRAFIA

I2E: MDA en I2e, http://www.i2e.com.es/blog/?p=218.

LCC – LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN: MDA Doctorado, http://www.lcc.uma.es/~canal/sabc/MDA-doctorado.pdf.

MODEL-DRIVEN ARCHITECTURE IN PRACTICE: A Software Production Environment Based on Conceptual Modeling. Oscar Pastor, Juan Carlos Molina.

MARTIN FOWLER: Martin Fowler on MDA, http://martinfowler.com/bliki/ModelDrivenArchitecture.html.

MODEL-DRIVEN SOFTWARE DEVELOPMENT: Model-Driven Software Development, Sami Beydeda · Matthias Book · Volker Gruhn (Eds.). Ed: Springer.