EL MODELADO CONCEPTUAL EN UNA ARQUITECTURA DIRIGIDA POR MODELOS

Ing. María Fernanda Granda Juca
Docente de la Facultad

Resumen

El objetivo de este artículo es explicar brevemente la importancia que tiene el modelado conceptual como actividad en el proceso de desarrollo de un sistema de información, más aún si se lo hace desde un enfoque de una Arquitectura Dirigida por Modelos (Model Driven Architecture - MDA).

Este trabajo describe cómo ha surgido el concepto de Modelado Conceptual a través de una síntesis histórica de su aporte en el ámbito del desarrollo de los sistemas de información. Además, se comentan los principios básicos en que se sustenta MDA, sus niveles de especificación, las tecnologías que le dan soporte y cómo ha generado un cambio paradigmático en la producción de software, donde el ente de valor ya no es el código sino el modelo en el cual está basado la generación del código, esta generación se puede realizar para varias plataformas, es decir se requiere modelar una vez y se puede generar el código varias veces e incluso hacia diferentes plataformas, de manera que el modelo llega a convertirse en código del software y cualquier cambio se debe realizar a este nivel de abstracción, al nivel de modelado conceptual.

¿Qué es Modelado Conceptual?

En el área del desarrollo del software, se utiliza el término modelado conceptual para definir la tarea de elicitar y describir los requisitos que un determinado sistema de información debe cumplir, tarea que es una parte importante en la Ingeniería de Requisitos, la cual es la primera fase del ciclo de desarrollo de un software. La especificación de este conocimiento es llamada *ontología* del dominio [1].

El primer trabajo en Modelado Conceptual ha sido registrado en 1958 [2], y trata sobre la formulación abstracta de los problemas de procesamiento de datos desarrollado por Young and Kent. Bubenko en su artículo [2] resume la evolución de la investigación y práctica en esta área durante un poco más de cuatro décadas. Este trabajo cubre los temas de investigación de los años 60s, donde surgieron ideas y conceptos pioneros tales como "los problemas de procesamiento de datos expresados formalmente de una manera independiente de la máquina", el desarrollo del reporte del grupo CODASYL "un Algebra de Información" en 1962, continúa con la introducción del método infológico y mensajes elementales de Langefors en 1965. Más tarde, la década de los 70s, se caracterizó por la introducción de nuevos modelos así como el refinamiento y extensión de una serie de lenguajes de modelamiento de la información como los modelos binarios, relacionales, modelos de datos semánticos, lenguaje lógico, modelamiento rol-objeto, entre otros. En los 80s,

la búsqueda estuvo centrada en encontrar un marco de trabajo común para mejorar la comprensión y optimizar los métodos y herramientas existentes, en este tiempo también se investigó como mejorar la fase de captura y validación de requerimientos del ciclo de vida de los sistemas, también se dieron las bases de los datos multitemporales, así como de los lenguajes orientados a objetos. Más tarde en los 90s el desarrollo y aplicación de métodos y técnicas más avanzadas de modelamiento conceptual continuaron extendiendo la discusión de los principios y problemas de investigación relacionados a los sistemas interoperables, heterogeneidad semántica, requerimientos no funcionales, modelos de negocio y modelos ontológicos. En este punto el modelamiento conceptual en el desarrollo de los sistemas de información es visto como un enfoque para capturar descripciones informales e inexactas de las necesidades de los usuarios, y luego de alguna manera transformarlas en algo completo, formal y consistente a través de las especificaciones conceptuales. Por lo tanto llega a ser necesario dar soporte a la participación de las partes interesadas y usuarios del sistema, y realizar el análisis y especificación de los requerimientos del negocio.

El aporte del Modelado Conceptual en MDA

A inicios de este siglo, la investigación en el área del modelado conceptual se centró en dar los primeros pasos para estandarizar los esfuerzos, así como también en el desarrollo de lenguajes y modelos ontológicos para dominios específicos. En el 2001, MDA surgió como las normas de la arquitectura base de OMG (Object Management Group) [3]. El software basado en MDA comienza con un modelo independiente de la plataforma (Platform Independent Model - PIM) que refleja el comportamiento y funcionalidad del negocio de la aplicación, este modelo se conoce como ***Esquema Conceptual***, y es construido usando un lenguaje de modelado basado en MOF (Meta Object Facility) de OMG, el más conocido es UML (Lenguaje Unificado Modelado). Las herramientas de desarrollo de MDA, convierten el PIM primero a un modelo de Plataforma Específica (Platform Specific Model - PSM) y luego a una aplicación de código en prácticamente cualquier plataforma middleware (Servicios Web, XML/SOAP, EJB, C#/.Net, CORBA de OMG, entre otras) [4] (Ver figura 1). De esta manera se cumple con los cuatro objetivos de MDA que son portabilidad, interoperabilidad, mantenibilidad y reusabilidad a través de la separación arquitectónica de modelos [5].

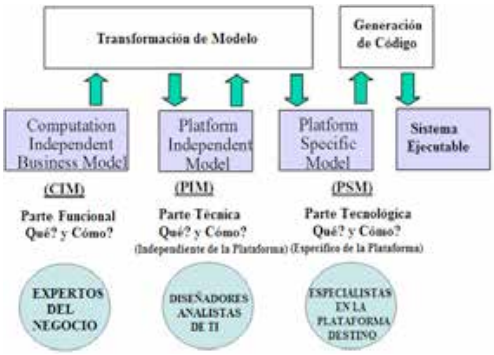

Figura 1. Niveles de especificación de servicios en la infraestructura MDA de la OMG.

En MDA, el modelado conceptual está fuertemente relacionado con la calidad final del producto del software, ya que los requisitos proveen las directrices para el diseño de la arquitectura del software, definen los objetivos del software y establecen las bases para medir la calidad del producto final [5].

Usos de los modelos del Software

La promesa de MDA es la de ordenar y facilitar el desarrollo del software en base a

reglas consistentes, auxiliada por un diseño visual, integrado, que es capaz de convertirse en código, permitiendo flexibilidad a largo plazo del proceso de desarrollo del software, para esto soporta cada fase del proceso de desarrollo, de la siguiente manera:

- Requisitos: especifica la estructura y comportamiento del sistema. Si el software ya existe, el hacer una reingeniería para obtener el modelo en el cual está basado el sistema, ayuda a comprender como está hecho.
- Implementación automática: genera automáticamente el código libre de errores, basado en la consistencia de los objetos integrados y permitiendo realizar un seguimiento hasta su implementación.
- Pruebas y simulación: facilita la detección de errores en las primeras fase del proceso de desarrollo del software, a nivel del diseño, minimizando costes y recursos. El tener modelos permiten construir un prototipo y simular el comportamiento para inferir, demostrar y validar propiedades a priori, conocer que tan fácil o difícil va a ser evolucionarlo o mantenerlo antes de crearlo, la validación se hace frente a los requisitos y se puede probar contra diversas infraestructuras.
- Mantenimiento: suministra facilidad, ya que al disponer del diseño en una forma legible por la máquina, cualquier cambio se debe hacer al nivel de modelo, esto permite mantener consistente lo que se modela con lo que se implementa.
- Integración: facilita la automatización de la producción de puentes de integración de datos y la conexión con las infraestructuras existentes (sistemas legados, sistemas de terceros).

En todas estas fases se facilita la comprensión del sistema a desarrollar, el control de cambios, el control de versiones con una documentación coherente y actualizada, el análisis de impacto, la utilización de nombres y definiciones únicas y coherentes, la rapidez de entrega de código, aumenta la calidad, la disminución de duración de los ciclos de desarrollo, por tanto mejora la productividad del proceso de desarrollo del software, facilita el reuso, la mantenibilidad y la reimplementación en otras tecnologías.

En este marco, MDA puede usarse sin dificultades en un método ágil, dejando a salvo por supuesto el hecho de que no se tocará el código fuente, como se hace normalmente en un enfoque ágil, donde se usa un generador automático de código como Oracle, Genexus, o algún otro; sino que se trabaja a nivel de un modelo de alto nivel, abstracto, donde los cambios son a nivel conceptual.

Tecnologías que dan soporte a MDA

El desarrollo y la implantación de herramientas que den soporte a los elementos de un entorno MDA (editor visual, repositorio de modelos, transformador modelo a modelo, transformador modelo a texto), son el requisito fundamental para que el Desarrollo de Software Dirigido por Modelos (DSDM/MDD) tenga éxito y se considere una realidad. MDA es la infraestructura para DSDM. En este contexto IBM, Borland y Microsoft están liderando la construcción de herramientas avanzadas para facilitar la implantación y el desarrollo del DSDM, entre ellas se tienen: IBM Rational Software Architect, Borland Together y Sparx Systems Enterprise Architect; otras conocidas son: ArcStyler, Acceleo, WebRatio, Mendix e Integranova.

Adicionalmente, existen herramientas que dan soporte por separado a cada uno de los elementos de un entorno MDA (ver figura 2):

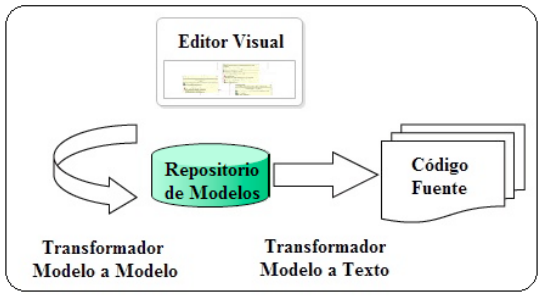

Figura 2. Elementos de un entorno MDA

Editor Visual de Modelos, para realizar el diseño e implementación de editores visuales que permitan gestionar los modelos: EMF, GEF+Draw 2D, GMF, herramientas DSL (Model Designer), DSL textuales (xText, oAW), METAEDIT+, entre las más conocidas.
Transformador Modelo a Modelo, para ejecutar la transformación de modelo a modelo: MTF, ATL, QVT, xTend - oAW, VIATRA2, AGG, entre las más divulgadas.
Repositorio de Modelos: para definir lenguajes de modelado y repositorio de modelos: MOF, "Profiles" de UML, EMF (eCore), Kermeta, ATLAS MegaModel Management (AM3), herramientas DSL (Domain Model Editor), XMI, entre las principales.
Transformador de Modelos a Texto, para generar el código fuente: programación manual, JET, XSLT, Lenguajes de Plantillas (FreeMarker, Velocity, DSL Tools, MOF-2Text, MOFScript, xPand-oAW), entre las principales.

Se puede optar por uno de los tres modelos de negocios en cuanto a la adquisición de las herramientas: código abierto, adquisición de la licencia o pagar por el uso del generador del código de la aplicación.

Dificultades de usar MDA

Existen muchas concepciones equivocadas acerca de lo que realmente es MDA, cómo y cuándo usarla, lo que genera dificultades al momento de querer optar por ella, algunas de estas dificultades se detallan a continuación:

- No se puede modelar cualquier tipo de aplicación, hay ciertas restricciones debido a la falta de madurez de las herramientas actuales.
- Modelar el software no es una solución inmediata a todos los problemas de desarrollo en una empresa, en realidad el desarrollo de software debe estar soportado por un proceso de desarrollo donde MDA puede estar integrada como marco de trabajo.
- No se puede generar 100% el código de la aplicación, ya que depende del tipo de aplicación. Algunas aplicaciones tipo CRUD (create, read, update and delete) se puede generar completamente, en otras se requiere algo de programación manual, esto depende de las capacidades de la herramienta y en esos casos hay que ser cuidadosos en marcar las zonas modificadas, ya que en una regeneración de código pueden perderse.
- Los desarrolladores de software basados en este nuevo paradigma, requieren formación y experiencia para poder centrar sus esfuerzos en todo lo que conlleva MDA, esta preparación tiene una curva de aprendizaje alta; además, MDA implica nuevos roles, tareas y dependencias en el equipo de desarrollo, y al principio esto afectará en la disminución de su productividad.
- No es suficiente comprar una buena herramienta MDA, sin considerar las características en cuanto al uso final de la misma: usabilidad, control de versiones, gestión de modelos, flexibilidad de la herramienta, capacitación del personal que la va a usar, etc.

Conclusiones

La mejor estrategia para desarrollar un software depende de muchos factores, entre ellos el tipo de aplicación que se va a desarrollar, los conocimientos del grupo de desarrolladores, el grado del cambio que se quiera dar, no hay ninguna receta mágica, sólo hay consejos que pueden ayudar a decidir.

El concepto de una herramienta L-CASE (Lower Case) generadora de código que hasta ahora se había conocido y usado para las fases de construcción e implantación de código, ahora es reemplazado por la iniciativa MDA de OMG. MDA parte de un diseño visual en UML, que es un modelo conceptual, abstracto, que luego es aplicado a una plataforma específica, al ser transformado de un modelo a otro, para finalmente a través de otra transformación obtener la generación del código ejecutable, de esta manera MDA da soporte al proceso de desarrollo del software, cubriendo todo o casi todo el ciclo de vida del producto, desde los requerimientos, el diseño conceptual, hasta la generación del código y el empaquetado para su despliegue. MDA puede integrarse a cualquier tipo de proceso de desarrollo de software, por ejemplo, en un proceso de tipo ágil se denomina Agile MDA.

El uso de MDA se justifica por la productividad que se alcanza en el desarrollo del software, especialmente en las factorías de software y también por la protección de la inversión frente a los continuos cambios en las tecnologías, ya que desde el modelo se puede evolucionar a una plataforma más actual; sin embargo, hay que tener presentes las dificultades que se tienen al usarla.

MDA es un área que ha despertado el interés de muchos investigadores de la comunidad del software, como el grupo MDA de la OMG, así como grupos de investigación de universidades, quienes están aportando con nuevas formas de automatizar la derivación de esquemas conceptuales en base a los requisitos de los interesados y usuarios del sistema, desarrollando herramientas y metodologías para compilar, optimizar y validar las transformaciones de los modelos a los diferentes niveles de abstracción, buscando formas de integrar modelos, generando pruebas de software basadas en modelos; es decir, investigando soluciones tecnológicas cada vez más potentes y maduras para hacer efectivo el Desarrollo de Software dirigido por Modelos; de manera que la visión de MDA “sólo es necesario y suficiente definir el esquema conceptual para desarrollar un Sistema de Información” se haga realidad.

Referencias Bibliográficas

[1]	*A. Olivé, Conceptual Modeling of Information Systems, New York: Springer, 2007.*
[2]	*J. Bubenko, From Information Algebra to Enterprise Modelling and Ontologies – a Historical Perspective on Modelling for Information Systems, Sweden: Springer, 2007.*
[3]	*J. M. J. Miller, «MDA Guide Version 1.0,» OMG, 1 05 2003. [En línea]. Available: http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf. [Último acceso: 10 09 2012].*
[4]	*OMG, «MDA Specifications,» 12 06 2003. [En línea]. Available: http://www.omg.org/cgi-bin/doc?omg/03-06-01. [Último acceso: 20 09 2012].*
[5]	*O. M. J. Pasto r, Model-Driven Architecture in Practice. A Software Production Environment Based on Conceptual Modeling, New York: Springer, 2007.*

María Fernanda Granda Juca, Ingeniera de Sistemas(1999, Universidad de Cuenca), Máster en Gerencia de Sistemas de Información (2009, Universidad de Cuenca), Profesora Principal tiempo completo de la Facultad de Ingeniería de la Universidad de Cuenca (1999-), Coordinadora del Centro de Capacitación en TICs de la Universidad de Cuenca (1999-2009), Directora de Escuela de la Escuela de Informática de la Universidad de Cuenca(2005-2008), Directora de la Maestría en Gerencia de Sistemas de Información de la Universidad de Cuenca (2010-2011), Becaria del Senescyt de la convocatoria 2011. Actualmente es investigadora del Centro de Investigación en Métodos de Producción de Software de la Universidad Politécnica de Valencia (2011-), donde está desarrollando su tesis de PHD en Informática en la área de Ingeniería del Software, los temas de interés en su investigación son Calidad & Pruebas del Software, Modelamiento Conceptual, Arquitectura Dirigida por Modelos e Ingeniería de Requisitos. Email: fernanda.granda@ucuenca.edu.ec / fgranda@pros.upv.es. Valencia, España.

“ Hardy Cross… ha puesto el dedo en la llaga: como la obligación del ingeniero es dar servicio a la humanidad y la del político también, es muy frecuente que el mérito de las obras se lo arrebaten los políticos, sobre todo cuando interviene la politiquería, estos llevan las de ganar. Pero el ingeniero no debe acobardarse ante esta situación que no es ni nueva ni especial; su objetivo debe ser siempre la obra misma y el fruto que ella rinde, en tanto que la gloria, por legítima que sea, inevitablemente debe estar subordinada. Más aún, al desarrollarse cada vez más el trabajo en equipo, la fama individual tiende a desaparecer, y la celebridad queda reservada a los pocos mortales que, por la naturaleza misma del trabajo que realizan, pueden sobresalir en aquello que indefectiblemente está ligado a la persona y a la personalidad de quien lo desempeña “.