Reflexiones acerca de la adopción de enfoques centrados en modelos en el desarrollo de *software*[1]

Reflections on the Adoption of Model-Based Approaches for Software-Development[2]

Reflexões sobre a adoção de enfoques centrados em modelos no desenvolvimento de *software*[3]

Juan Bernardo-Quintero[4]
Jhon Freddy Duitama-Muñoz[5]

[1] Fecha de recepción: 21 de septiembre de 2010. Fecha de aceptación: 17 de enero de 2011. Este artículo se deriva de un proyecto de investigación financiado por el Centro de Excelencia Alianza Regional en TIC Aplicadas (ARTICA), contrato Colciencias 584 del 2008, Universidad de Antioquia, Medellín, Colombia.

[2] Submitted on: September 21, 2010. Accepted on January 17, 2011. This article is the result of the research project financed by the Excellence Research Center Regional Alliance on Applied Information and Communication Technologies (ARTICA), with Colciencias registration number 584/2008, Universidad de Antioquia, Medellin, Colombia.

[3] Data de recepção: 21 de setembro de 2010. Data de aceitação: 17 de janeiro de 2011. Este artigo é derivado de um projeto de pesquisa financiado pelo Centro de Excelência Aliança Regional em TIC Aplicadas (ARTICA), contrato Colciencias 584 de 2008, Universidade de Antioquia, Medellín, Colômbia.

[4] Ingeniero. Doctor en Ingeniería, Universidad de Antioquia, Medellín, Colombia. Docente del Departamento de Ingeniería de Sistemas, investigador del Grupo de Investigación en Ingeniería y *Software,* Universidad de Antioquia. Correo electrónico: jbquintero@udea.edu.co.

[5] Ingeniero de sistemas. Doctor en Informática, Institut National des Télécommunications, París, Francia. Docente del Departamento de Ingeniería de Sistemas y director del Grupo de Investigación en Ingeniería y Software, Universidad de Antioquia, Medellín, Colombia. Correo electrónico: freddy.duitama@udea.edu.co.

Resumen

La reutilización de modelos es una de las estrategias de mayor acogida en las recientes propuestas metodológicas en desarrollo de *software*, al punto de involucrar de forma exhaustiva el concepto de modelo y prometer que para construir una aplicación de *software* basta con construir modelos y transformarlos de forma semiautomática y asistida en el código de un sistema de información. Sin embargo, aún existen grandes retos que afrontar en la adopción de enfoques centrados en modelos, por ejemplo, una gran diversidad de técnicas, lenguajes y herramientas para transformar modelos. Esto deriva en una falta de unificación que les dificulta a los equipos de desarrollo iniciar un proyecto con la certeza de estar usando las estrategias de transformación apropiadas para su proyecto o empresa. Este trabajo escudriña en los diferentes planteamientos de la ingeniería de modelos y en los estudios previos en transformación de modelos, para servir de referencia en la adopción de enfoques centrados en modelos en el desarrollo de *software*.

Palabras clave

Desarrollo de *software* de aplicación, arquitectura dirigida por modelos, ingeniería de *software.*

Abstract

Reusing models is one of the most widely accepted strategies among recent methodological approaches to software development. It is so much so that these approaches have extensively included various models with the suggestion that, in order to develop a software application, it is just necessary to build models and to transform them semi-automatically with the support of an information system. However, there are still considerable challenges in the adoption of model-focused approaches, such as a variety of techniques, languages, and tools available for model transformation. This variety of resources complicates the standard selection of strategies for transforming models which are appropriate for a particular project or company. This paper analyzes the different model engineering approaches, as well as previous studies in model transformation, in order to serve as a guide in the selection process of model-focused approaches for software development.

Key words

Application software – development, model-driven software architecture, software engineering.

Resumo

A reutilização de modelos é uma das estratégias com maior receptividade nas recentes propostas metodológicas em desenvolvimento de *software*, ao ponto de envolver de forma exaustiva o conceito de modelo e prometer que para construir uma aplicação de *software* basta construir modelos e transformá-los de forma semi-automática e assistida no código de um sistema de informação. Contudo, ainda existem grandes desafios que enfrentar na adoção de enfoques centrados em modelos, por exemplo, uma grande diversidade de técnicas, linguagens e ferramentas para transformar modelos. Isto deriva em uma falta de unificação que traz dificuldades às equipes de desenvolvimento iniciar um projeto com a certeza de estar usando as estratégias de transformação apropriadas para seu projeto ou empresa. Este trabalho esquadrinha as diferentes propostas da engenharia de modelos e os estudos prévios em transformação de modelos, para servir de referência à adoção de enfoques centrados em modelos no desenvolvimento de *software*.

Palavras chave

Desenvolvimento de *software* de aplicação, arquitetura dirigida por modelos, engenharia de software.

Introducción

La ingeniería dirigida por modelos (*Model Driven Engineering* [o MDE]) define los mecanismos para utilizar modelos en la automatización de tareas propias del desarrollo de *software*, por ejemplo: configuración, análisis y diseño, generación de código, refinamiento, refactoría, traducción a otros lenguajes o plataformas, etc. Los planteamientos de MDE hacen hincapié en aspectos como la construcción de modelos en exhaustiva comunicación con los usuarios, para que tengan sentido desde su punto de vista; adicionalmente, sugiere que dichos modelos sirven como base para poner sistemas en ejecución.

Sus inicios se remontan a los años ochenta, cuando en el marco de la ingeniería de *software* (Somerville, 2005) aparecieron las primeras herramientas tipo *Computer Aided Software Engineering* (CASE). En la década de los noventa, el auge de los modelos y la aparición de *Unified Modeling Language* (UML) incentivaron su uso (Booch et al., 2002; Object Management Group, 2010). Recientemente, la comunidad mundial de desarrollo de *software* ha empezado a volcar su atención en la MDE. Por esto, en muchas de las siglas que rigen el desarrollo de *software* en la actualidad aparecen las letras MD para referirse a la frase en inglés *model driven*, como es el caso de la famosa MDA que, junto con otras siglas, fue registrada como marca por parte del Object Management Group (2003). La MDA propone un proceso de desarrollo de *software* basado en modelos, que define las características básicas de los modelos que se van a construir en cada una de sus etapas (Quintero y Anaya, 2007).

Uno de esas siglas registradas por el Object Management Group fue la de Model Driven Development (MDD), que es una aproximación de desarrollo de *software* centrada en crear modelos para describir los elementos de un sistema. Su objetivo fundamental es aumentar la productividad, maximizando compatibilidad entre los sistemas, simplificando el proceso del diseño y promoviendo la comunicación entre los individuos y los equipos que trabajan en el sistema (Selic, 2003). La sigla principal que aún no ha sido registrada por

Object Management Group es MDE. En consecuencia, es esta la sigla usada actualmente por la comunidad investigadora internacional cuando se refieren a ideas relacionadas con la ingeniería de modelos sin centrarse exclusivamente en los estándares y planteamientos de Object Management Group.

Por otro lado, el *Model Driven Software Development* (MDSD) es una propuesta impulsada por algunas industrias, firmas de consultoría y universidades. Trata el uso de lenguajes específicos de dominio o (*Domain Specific Languages* [DSL]) para crear modelos que expresen estructura o comportamiento de la aplicación de una forma eficiente y específica del dominio. Estos modelos son transformados posteriormente en código ejecutable por una secuencia de transformación de modelos.

Para simplificar y dar claridad respecto a la diversidad de enfoques dirigidos por modelos, se plantea la siguiente expresión: MDE ≅ MDD ≅ MDSD ⊃ MDA. Esta ilustra cómo en el marco del desarrollo de *software* MDE, MDD y MDSD son aproximaciones de desarrollo similares; mientras que, de acuerdo con los planteamientos de MDSD (Völter y Stahl, 2006), MDA es una de las modalidades para aplicar MDSD, en el que se utiliza UML como lenguaje de modelado y el DSL puede ser definido utilizando perfiles.

Al analizar los proyectos que involucran *Business Process Management* (BPM), se encuentra que estos también se ajustan a la definición de MDE o MDD, puesto que en muchos casos la implementación de los procesos de negocio conlleva la construcción de *software*, que es generado a partir de modelos construidos usando *Business Process Modeling Notation* (BPMN) (Object Management Group, 2009), que son ejecutados en un *Business Process Management System* (BPMS). Este es el motivo que justifica la inclusión del desarrollo de *software* basado en BPM como un enfoque centrado en modelos.

Adicional a MDA, MDSD y el desarrollo en proyectos basados en BPM, existen otros enfoques para la construcción de *software*, por ejemplo: las líneas de productos de *software* (*Software Product Lines* [SPL]) (Northrop, 2008); las factorías de *software* (Greenfield y Short, 2004), y la orientación por aspectos (Chitchyan y otros, 2005). Esta proliferación de enfoques genera una problemática que deben afrontar los equipos de trabajo antes de iniciar un proyecto de construcción de *software* que involucre nuevos enfoques de desarrollo: ¿cuáles de estos es más apropiado para mi proyecto o empresa? ¿Qué beneficios, retos y problemas debo afrontar en la adopción de un nuevo enfoque de desarrollo? ¿Cuáles son los puntos de especial atención?

Las reflexiones presentadas en este artículo tienen como objetivo proporcionar a los equipos de trabajo herramientas y argumentos que respalden la toma de decisiones respecto a la adopción de enfoques de desarrollo de *software* centrado

en modelos. Por tal motivo se pone el relieve en aquellos enfoques que usan modelos de manera exhaustiva, como son MDA, MDSD y los basados en BPM. Para tal propósito, este artículo se organiza de la siguiente manera: la sección uno explora los pros y los contras de la ingeniería dirigida por modelos, la sección dos muestra las problemáticas de la transformación en los enfoques centrados en modelos, la sección tres revisa los estudios acerca de las técnicas y los lenguajes de transformación de modelos, la sección cuatro examina los estudios previos acerca de las herramientas de transformación de modelos y la sección cinco plantea unas conclusiones y esboza trabajos futuros al respecto.

1. Pros y contras de la ingeniería dirigida por modelos

Se han realizado diversos trabajos tendientes a dictaminar cómo se están viendo afectadas las comunidades de *software* con la adopción de MDD. En dichas investigaciones se pone en evidencia que, a pesar de las alentadoras fortalezas, existen falencias que deben subsanarse. La compañía de investigación de mercado Forrester Research Inc. (2008) ha realizado estudios en los cuales resalta algunas fortalezas del MDD y afirma que ayuda a las organizaciones de desarrollo a obtener el control de la documentación y otros artefactos existentes, mejora la productividad e, incluso, después de un período de lanzamiento inicial, se hace más fácil para los equipos de desarrollo actualizar y mantener aplicaciones. Todo esto se logra una vez ha sido introducida suficiente información acerca de los tipos de desarrollo y las plataformas en las herramientas MDD, de tal forma que se le permita al usuario desarrollador describir el tipo de detalles que deben aplicarse. Este estudio plantea los beneficios de la madurez del desarrollo dirigido por modelos como lo ilustra la Tabla 1.

Tabla 1. Beneficios en la madurez de MDD

Nivel	Beneficios
Desarrollo apoyado por modelos (*Model-Aided Development*)	Si también se adopta en el frente de negocio (BPM), apoya la alineación negocio/TI sobre los requisitos
Desarrollo asistido por modelos (*Model-Assisted Development*)	Disminuye el costo de los proyectos Mejora la trazabilidad de los requisitos Incrementa la productividad Puede mejorar la alineación negocio/TI
Desarrollo dirigido por modelos (*Model-Driven Development*)	Disminuye el *time-to-market* de los productos Mejora la calidad de los diseños y arquitectura del *software* Consolida el puente entre negocio y TI

Fuente: presentación propia de los autores con base en (Forrester Consulting, 2008).

En contraposición a estos beneficios, aparece un estudio presentado en el Taller Internacional sobre Modelos en Ingeniería de Software, en el 2008, publicado por la *Association for Computing Machinery* (Forward y Lethbridge, 2008), en el que se encuentra un *ranking* con los posibles problemas en la aplicación de los enfoques centrados en modelos, ordenado de acuerdo con el porcentaje de aceptación efectiva como problema por parte de los desarrolladores. Según este estudio, el problema que más les preocupa a los desarrolladores es lo obsoleto que se puede volver un modelo cuando el código se cambia manualmente o lo obsoleto que se puede volver el código de sistemas en producción cuando los modelos son modificados. En la Tabla 2 se listan los cinco primeros en el *ranking*, con su porcentaje de aceptación por parte de los desarrolladores.

Tabla 2. Problemas en el desarrollo centrado en modelos

	Descripción del Problema	Aceptación (%)
1°	Los modelos se vuelven obsoletos e incompatibles con el código	68,5
2°	Los modelos no se pueden intercambiar fácilmente entre las herramientas	51,6
3°	Las herramientas de modelado son "peso pesado" al instalar, aprender, configurar y utilizar	39,1
4°	El código generado por una herramienta de modelado no es del tipo que me gustaría	38,5
5°	No se puede describir el tipo de detalles que deben ser implementados	36,0

Fuente: presentación propia de los autores con base en (Forward y Lethbridge, 2008).

Estos problemas han afectado en gran medida el ámbito de las herramientas que apoyan los enfoques centrados en modelos. Vemos, por ejemplo, cómo algunas de las herramientas que inicialmente parecían bastante prometedoras en el marco del MDA, como ArcStyler y OptimalJ (García et al., 2004), que incluso estaban incluidas en el listado oficial del Object Management Group, han perdido su auge, al punto de que en algunos casos las compañías encargadas tuvieron que cambiar de estrategia de negocio frente al tema.

Uno de los aspectos que pueden influir negativamente lo constituye la imposibilidad de generar para diversas plataformas, pues OptimalJ permitía generar sólo a plataformas Java; mientras que en ArcStyler se necesitaban desarrollar o conseguir complejos cartuchos que permitían la generación a plataformas diferentes a las iniciales. Los cartuchos son componentes desplegables que encapsulan una función de transformación, y se le pueden instalar a la herramienta para agregarle funcionalidad. Otro factor negativo lo constituyen las pocas posibili-

dades para definir, por ejemplo, la apariencia (*look and feel*) de la aplicación final, las estrategias de interacción humano computador o las particularidades del código generado. No considerar la existencia de algunos de los problemas mencionados puede dar al traste con la ejecución de proyectos que busquen adoptar enfoques centrados en modelos; por esto es importante revisar los principales motivos para que proyectos de adopción de enfoques dirigidos fracasen. La Tabla 3 muestra ocho razones por las cuales un enfoque dirigido por modelos puede fallar y cuáles consideraciones se deben tener para evitar el fracaso (Den Haan, 2008).

Tabla 3. Motivos para el fracaso de enfoques dirigidos por modelos

Descripción del motivo	Consideraciones para evitar el fracaso
No orientarse a los objetivos de MDE a corto y a largo plazo	Tener en cuenta los objetivos de MDE a corto plazo que apoyan la construcción de artefactos, y tener en cuenta los objetivos a largo plazo que buscan alargar el tiempo antes de que los artefactos sean obsoletos
Concentrarse en una sola dimensión de modelado	Poner atención a las dos dimensiones de modelado: la del dominio o área de aplicación y la de la tecnología
Enfocarse sólo en generar artefactos nuevos	Considerar la generación de artefactos que se integran con artefactos existentes o que han sido modificados luego de generarse
Usar sólo lenguajes de propósito general como UML	Cubrir todos los dominios necesarios, incluso los que no son cubiertos por lenguajes de propósito general. Por ejemplo, UML no cubre el modelado de la interfaz de usuario (Balmelli et al., 2006)
Abusar de diversos DSL personalizados	Evitar la proliferación de diversos DSL, para no complicar el aprendizaje de los lenguajes, ni la gestión de modelos
Transformar modelos no ejecutables plenamente	Evitar especificación de transformaciones ambiguas o elementos innecesarios en los modelos
No probar los modelos construidos	Realizar pruebas tendientes a determinar la calidad de los moldeos construidos (Mohagheghi y Aagedal, 2007)
No disponer de suficientes herramientas adecuadas	Respaldar en herramientas gran parte de las labores de MDE para potenciar la automatización. Por ejemplo, la transformación de modelos, generación de código, la traducción a otros lenguajes, etc.

Fuente: presentación propia de los autores con base en (Den Haan, 2008).

Estos y otros beneficios y problemas del desarrollo dirigido por modelos definen importantes retos por afrontar, que están siendo investigados por la comunidad informática en general. Un ejemplo de estas investigaciones es el proyecto Modelplex (Information Society Technologies [IST], 2006), cofinanciado por la Comisión Europea en el marco del Programa IST, el cual plantea la

siguiente clasificación de los retos que se deben enfrentar en el marco de MDE (Frabce y Rumpe, 2007):

- Retos de los lenguajes de modelado: surgen de preocupaciones asociadas a respaldar la creación y uso de abstracción en los lenguajes de modelado y apoyar el análisis riguroso de los modelos; por ejemplo, complementar los modelos con formalismos que permitan especificar restricciones y transformaciones.
- Retos de separación de asuntos (*Separation of Concerns* [SoC]): surgen de problemas asociados a modelar sistemas utilizando múltiples puntos de vistas que se superponen y que utilizan lenguajes posiblemente heterogéneos; por ejemplo, construir modelos lógicos y físicos o los detalles para permitir generar a plataformas, como J2EE, JEE, .Net o PHP.
- Retos de la manipulación y gestión de modelos: surgen de problemas asociados con (1) la definición, análisis y uso de transformaciones de modelos; (2) mantener vínculos de trazabilidad entre los elementos de los modelos para apoyar su evolución y de ingeniería de ida y vuelta; (3) transformar modelos manteniendo coherencia entre los puntos de vista de los diferentes implicados en el desarrollo; (4) el rastreo de versiones, y (5) el uso de modelos en tiempo de ejecución.

Este proyecto Modelplex (IST, 2006), que pretende desarrollar una solución abierta para mejorar la calidad y productividad de sistemas complejos, a la par que lidera su industrialización y garantiza el éxito de su adopción, también ha publicado estudios donde se plantean los tres principales retos de MDE (Mohagheghi et al., 2009):

- Soluciones para el dominio: cómo disponer de los ambientes de modelado, los lenguajes y las herramientas apropiadas para apoyar el modelado específico del dominio.
- Complejidad en el proceso de desarrollo: cómo tener el control sobre las diversas actividades del proceso, pero en entornos con herramientas adaptativas e integrables, desarrollo de lenguajes específicos de dominio, editores y generadores; mientras que a la par se adquiere la suficiente experiencia y se madura el proceso.
- Experiencia y capacitación: cómo tener acceso a expertos y formación para la adopción de enfoques centrados por modelos. Este tipo de problemas y retos también son enfrentados al momento de desarrollar aplicaciones usando enfoques basados en BPM. *El cuadrante mágico para los BPMS* es un estudio de Gartner Research frecuentemente referenciado (Hill et al., 2009),

el cual muestra que las fortalezas y debilidades de los diferentes BPMS para implementar los procesos de negocio giran en torno a este tipo de retos y problemas. Tal argumento le da validez a la inclusión del desarrollo basado en BPM, como un enfoque centrado en modelos.

2. Problemáticas de la transformación en los enfoques centrados en modelos

La transformación de modelos les permite a los desarrolladores construir modelos con la representación del problema y transformarlos sucesivamente hasta llegar a un sistema computacional que funcione. Esta promesa realizada en la mayoría de los recientes enfoques metodológicos potencia enormemente las posibilidades de reutilizar elementos propios del desarrollo de *software*, como son objetos, componentes, patrones y *frameworks* (Larman, 2005), para la generación casi automática de aplicaciones de *software*. Una de las estrategias típicas de transformación se fundamenta en construir un modelo basado en un metamodelo de origen al que se le aplican unas reglas de transformación definidas, y se transforma en un modelo que se basa en un metamodelo destino (Bézivin, 2004).

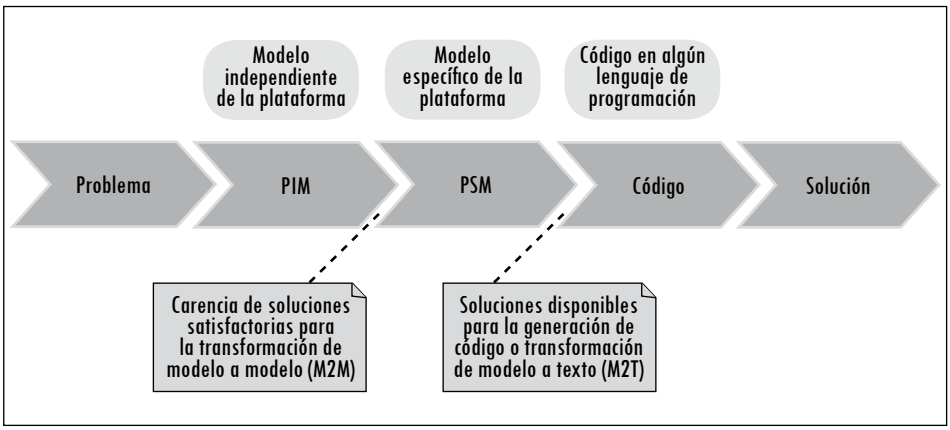
La transformación de modelos desempeña un papel de vital importancia en los enfoques centrados en modelos, al punto de haber sido señalada como "el corazón y el alma" del desarrollo de *software* dirigido por modelos (Sendall y Kozaczynski, 2003). Para verificar su importancia, se analizan las problemáticas de la transformación en MDA, MDSD y en los enfoques basados en BPM.

2.1 Consideraciones de la transformación en MDA

La transformación en MDA se usa principalmente para convertir un modelo independiente de la plataforma en uno o varios modelos específicos de la plataforma (Kleppe et al., 2003). Involucra un amplio conjunto de estándares del Object Management Group (2006, 2007b y 2009a), como una especificación de metamodelo llamado *Meta Object Facility* (MOF), un lenguaje de restricciones llamado *Object Constraint Language* (OCL) y un lenguaje de transformación llamado *Query/View/Transformation* (QVT).

Las soluciones y herramientas disponibles para MDA han evolucionado ampliamente en lo que concierne a la generación de código, situación contraria a la falta de soluciones satisfactorias para la transformación de modelo a modelo, puesta en evidencia en (Czarnecki y Helsen, 2006). La Figura 1 ilustra esta situación en el marco de la transformación en MDA.

Figura 1. Consideraciones para la transformación de modelos en MDA



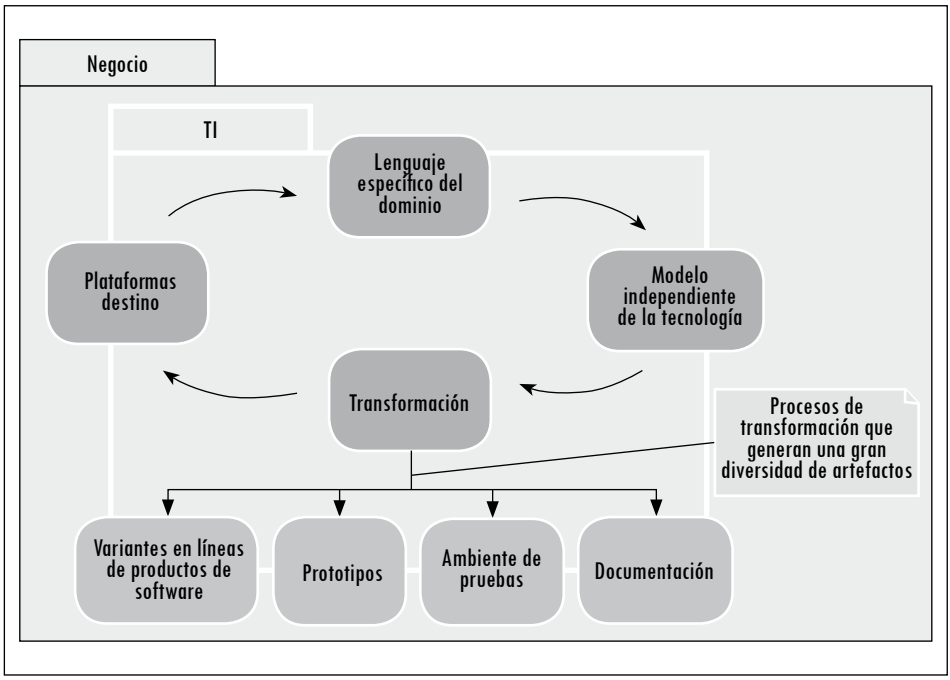
Fuente: presentación propia de los autores.

2.2 Consideraciones de la transformación en MDSD

El MDSD utiliza tres conceptos para definir la arquitectura de referencia de los sistemas que se van a desarrollar: la transformación, un lenguaje específico del dominio y una plataforma destino. Con base en estos el desarrollador construye un modelo generalmente independiente de la tecnología, el cual es transformarlo en diversos artefactos (Quintero y Pérez, 2009). Teniendo en cuenta que el MDA es un "aroma" de MDSD, las consideraciones de la transformación son también validas en MDSD. Sin embargo, los planteamientos propios del MDSD dan pie a la generación de un gran número de artefactos a partir de los modelos; por ello su gestión y manipulación es un frente en el que se debe poner especial cuidado (Frabce y Rumpe, 2007).

Para gestionar esta variedad de artefactos se necesitan herramientas de diversos tipos, por ejemplo: ambientes de prueba, documentadores, herramientas de transformación, editores de lenguajes específicos de dominio, entre otras. En este frente pueden surgir problemas de control sobre el proceso. La Figura 2 ilustra esta situación en MDSD.

Figura 2. Consideraciones para la transformación de modelos en MDSD



Fuente: presentación propia de los autores.

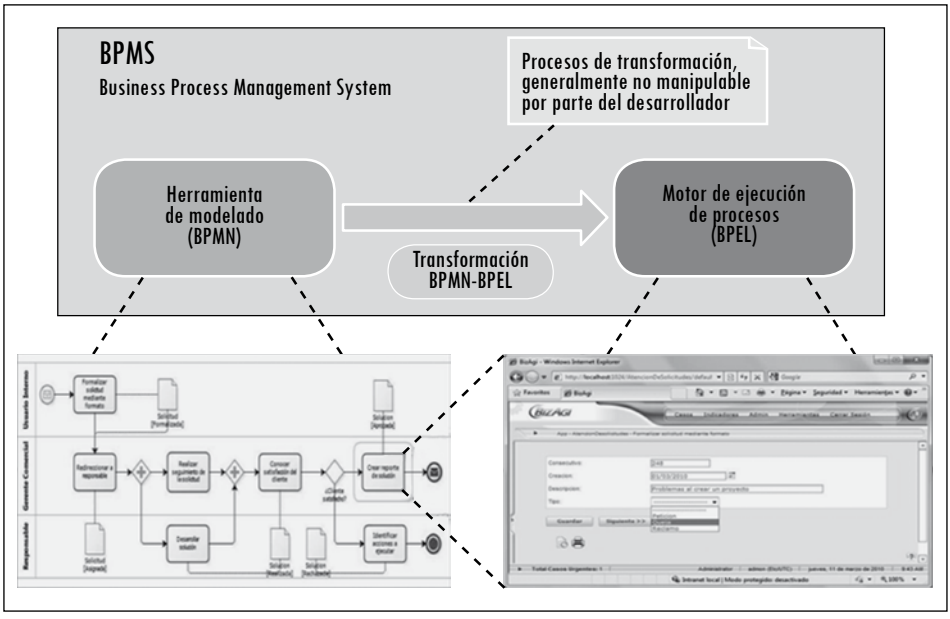
2.3 Consideraciones de la transformación en enfoques basados en BPM

Para conseguir ejecutar un proceso de negocio se construye un diagrama que lo represente usando BPMN. Así se complementan cada uno de los pasos del proceso con la información que los hace ejecutables, por ejemplo, los objetos de negocio relacionados y la disposición de sus ítems en los formularios. Luego estos procesos despliegan parte de sus tareas a través de una aplicación web que utiliza como plataforma un BPMS, el cual adicionalmente de permitir la ejecución de los procesos de negocio apoya su control y gestión.

Una de las consideraciones relevantes de la transformación en BPM la constituye revisar la intervención que puede tener el desarrollador en las transformaciones entre el BPMN y el *Web Services Business Process Execution Language* (WSBPEL), comúnmente llamado BPEL (OASIS, 2007). Por lo general esta transformación es potestad de los BPMS, y en gran medida se hacen transparentes. En algunos casos esta poca intervención en la transformación es considerada positiva, pues

disminuye las responsabilidades de los desarrolladores; sin embargo, cuando se quieren realizar algunos tipos de cambios en las aplicaciones que se van a generar, aparecen restricciones que terminan limitando las posibilidades de los desarrolladores en frentes como la definición de las estrategias de interacción humano-computador, el lenguaje de programación o las características del código generado. Esta es la razón por la cual algunos autores plantean que la transformación entre BPMN y BPEL debe cumplir características básicas como integridad, exactitud, legibilidad y reversibilidad (Dumas, 2009). Estos aspectos motivan el trato de la transformación entre BPMN y BPEL de forma similar a las transformaciones en MDA o en MDSD. La Figura 3 ilustra esta situación en el marco de BPM.

Figura 3. Consideraciones para la transformación de modelos en BPMN



Fuente: presentación propia de los autores.

La importancia que se le da a la transformación en el marco de MDE y las problemáticas de ésta en los enfoques centrados en modelos, ponen en evidencia que la transformación es un aspecto que se debe tener en especial consideración al momento de adoptar un enfoque de desarrollo centrado en modelos. Por tal motivo las dos próximas secciones revisan con detenimiento los estudios previos

que se han realizado en lo que concierne a la transformación de modelos; los cuales típicamente se han planteados en dos modalidades: estudios taxonómicos para tratar las técnicas y lenguajes de transformación, y estudios empíricos para tratar las herramientas de transformación de modelos.

3. Estudios taxonómicos acerca de las técnicas y los lenguajes de transformación de modelos

Esta sección muestra los análisis realizados mediante estudios taxonómicos de las técnicas y lenguajes de transformación de modelos, que revisan minuciosamente las estrategias utilizadas en la transformación de modelos, para plantear una clasificación basada en la forma en que dichas estrategias son implementadas. Estas taxonomías resultan de utilidad al considerar los enfoques centrados en modelos para el desarrollo de *software*, porque apoyan la resolución de un conjunto importante de interrogantes en el proceso de adopción de un enfoque: ¿cuántos pasos tendrán las transformaciones? ¿Cuál será el origen y destino de cada transformación? ¿Qué modalidades de transformación se van a utilizar? ¿Cuáles serán los mecanismos para la definición de las reglas? ¿Qué herramientas se utilizarán para la transformación?

Estos interrogantes han sido planteados de forma intencional, para coincidir con los criterios de clasificación utilizados en cada uno de los estudios taxonómicos ilustrados en la Tabla 4; de esta forma, las categorías planteadas en cada clasificación darán pie a las posibles respuestas de cada interrogante.

Tabla 4. Estudios taxonómicos alrededor de la transformación de modelos

Número de pasos en la transformación (Völter y Stahl, 2006; Haywood, 2004)	
Paso simple	Enfoque *elaboracionista* que parte de modelos ejecutables hasta en un 70% que no contienen toda la información necesaria para la generación de la aplicación. La información faltante es adicional al modelo intermedio o al código.
Múltiples pasos	Enfoque *traslacionista* que parte de modelos ejecutables 100%. Los compiladores de modelos tienen toda la responsabilidad de generar el sistema completo; generalmente este enfoque se apoya en lenguajes formales.
Origen y destino de la transformación (Mens y Van Gorp, 2006; Czarnecki y Helsen, 2006)	
M2M: modelo a modelo	Cambiar el grado de abstracción de un modelo para transformarlo en otro, llevándolo desde el espacio del problema hasta el espacio de la solución.
M2T: modelo a texto (M2C: modelo a código)	Generación del código de la aplicación de *software*. Se habla de generación de código cuando se generan sólo algunas de las capas de la arquitectura. Si se generan todas las capas y se dejan los archivos listos para su ejecución o compilación, se habla de generación de aplicaciones.

Continúa

Modalidad de transformación (Object Management Group, 2003)	
Basada en metamodelos	Se definen las reglas de transformación de los metamodelos de origen al metamodelo de destino; luego se transforman los modelos como instancias de los metamodelos.
Basada en marcas	El modelo de origen se marca con base en el metamodelo de destino, proporcionando la información adicional necesaria para guiar la transformación.
Basadas en tipos	Se especifican dos conjuntos de tipos, para el modelo de origen y el de destino, y la transformación se basa en la correspondencia entre estos conjuntos de tipos.
Basadas en patrones	Patrones predeterminados en el modelo de origen se transforman en patrones predeterminados en el modelo de destino.
Basada en mezcla de modelos	La transformación toma como entrada un modelo de origen y le adiciona uno o varios modelos, con objeto de producir un modelo de destino.
Mecanismos de definición de las reglas (Czarnecki y Helsen, 2006)	
Enfoques de manipulación directa	Son *frameworks* o *Application Programming Interface* (API) con una representación interna del modelo. Poseen funciones que pueden ser invocadas desde un programa y que se encargan de la transformación. Ejemplos: proyecto Jamda y *Java Metadata Interface* (JMI) (Sun Developer Network, 2002).
Enfoques relacionales	Enfoques declarativos de gran afinidad con las matemáticas. Sus expresiones relacionan elementos del origen y del destino. Libres de efectos laterales, suelen soportar "volver atrás". Ejemplo: QVT (Object Management Group, 2009a).
Enfoques basados en transformación de grafos	Aprovechan las técnicas y conceptos definidos en la teoría de grafos para aplicarlos en la transformación de modelos. En esta categoría aparecen lenguajes como BOTL, VIATRA, ATOM3, UMLX y GReAT analizados en (Mens y Van Gorp, 2006).
Enfoques dirigidos por estructura	Se dividen en dos fases: crear una estructura jerárquica para el modelo destino y poner atributos y referencias en el origen. Estos enfoques suelen generan aplicaciones a partir de un modelo de domino o modelo conceptual.
Enfoques híbridos	Combinan algunos de los demás enfoques planteados, por ejemplo, *Atlas Transformation Language* (ATL) de Eclipse, que puede utilizar reglas declarativas o imperativas (The Eclipse Foundation, 2006) o *eXtended Development Environment* (XDE) (IBM, 2004), que usa un lenguaje de restricciones y tiene implementación automática de patrones GoF (Gamma y otros, 1995).
Otros enfoques	Como *Common Warehouse Metamodel* (CWM), que provee un mecanismo para ligar los elementos del origen con los del destino, pero la derivación no está prescrita en CWM (Object Management Group, 2003b); o *eXtensible Stylesheet Language Transformations* (XSLT) (W3C, 1999), que sirve para transformar modelos serializados en XML, usando *XML Metadata Interchange* (XMI) (Object Management Group, 2007a).
Herramientas usadas para la transformación (Jézéquel, 2006)	
Lenguajes de programación de propósito general	Usan lenguajes convencionales para realizar la transformación, aunque no se necesita aprender un lenguaje adicional; puede ser difícil de aplicar para sistemas complejos.

Herramientas de transformación genéricas	Como XSLT que incluye plantillas para definir las reglas de transformación; o las herramientas de transformación de grafos que, según Jézéquel, son poderosas pero complejas, incluso requieren formación avanzada en el área de modelos.
Lenguajes de *script* en herramientas CASE	Permiten definir nuevas transformaciones y aunque actualmente son maduros, suelen usar lenguajes propietarios. Ejemplo: cartuchos en ArcStyler o AndroMDA (AndroMDA, 2006).
Herramientas de transformación de modelos	Dedicadas exclusivamente a la definición y ejecución de reglas de transformación. Suelen basarse en estándares como QVT o ATL, y usar lógicas declarativas o imperativas.
Herramientas de metamodelado	Sirven para construir los modelos de los modelos, y de esta forma apoyar la transformación. Generalmente usan diagramas de clases o equivalentes. Ejemplo: Kermeta (Drey y otros, 2010) o ATOM3 que adicionalmente usa transformación de grafos (De Lara, 2006).

Fuente: presentación propia de los autores.

Adicional a este conjunto de categorías que dan respuesta a importantes interrogantes en la adopción de un enfoque centrado en modelos en el desarrollo de *software*, estos estudios taxonómicos dejan otras conclusiones y lecciones aprendidas que resultan de utilidad: según (Mens y van Gorp, 2006), una de las posibles desventajas es que varias de las técnicas expresadas en estas taxonomías no son compatibles unas con otras. De acuerdo con (Czarnecki y Helsen, 2006), XSLT es una alternativa bastante atractiva; sin embargo, la elocuencia y pobre legibilidad de los modelos de transformación con XSLT conllevan problemas de escalabilidad que hacen de esta una alternativa poco mantenible. A pesar de esto las soluciones existentes para la época en la generación de código eran satisfactorias; no sucedía lo mismo con la transformación de un modelo a otro, pues existían pocos lenguajes de transformación populares y no había herramientas maduras de uso industrial reconocido.

4. Estudios empíricos acerca de las herramientas de transformación de modelos

Las herramientas de transformación de modelos se han analizado a través de estudios empíricos con el objetivo de determinar las características que estas deben tener. Tales características pueden ser utilizadas posteriormente como criterios para realizar la evaluación de un conjunto de herramientas. Los estudios empíricos resultan de utilidad en la adopción de enfoques centrados en modelos, porque les dan a los equipos de trabajo directrices acerca de cómo las herramientas están dando soporte a los planteamientos de los diferentes enfoques centrados en modelos.

Los estudios empíricos en transformación de modelos se han planteado generalmente en el marco de MDA. La Tabla 5 muestra los estudios empíricos más relevantes realizados alrededor de la transformación de modelos, con las particularidades de las evaluaciones realizadas en cada uno.

Tabla 5. Estudios empíricos alrededor de herramientas o lenguajes de transformación

Referencia	Institución	Particularidades de la evaluación
King's College London, University Of York (2003)	King's College London, University of York, Reino Unido	Cuantitativa, 16 características calificadas en una herramienta
García et al. (2004)	Universidad de Murcia, España	Cuantitativa, 17 características calificadas en 2 herramientas
Wang (2005)	Universidad de Tecnología de Viena, Austria	Cualitativa, 22 características revisadas en 6 herramientas o lenguajes
Akhter y Tariq (2005)	Royal Institute of Technology, Karolinska University Hospital, Suecia	Cuantitativa, 70 calificadas características en 9 herramientas
Grønmo y Oldevik (2005)	SINTEF, Noruega	Cualitativa, 14 características revisadas en 11 herramientas o lenguajes
Mens et al. (2006)	Universidad de Mons-Hainaut, Universidad Antwerpen, Bélgica	Cualitativa, 12 características revisadas en 4 herramientas
Quintero y Anaya (2007)	Universidad EAFIT, Colombia	Cuantitativa, 26 calificadas características en 10 herramientas
Calic et al. (2008)	Universidad de Nevada, Estados Unidos	Plantea 54 características, clasificadas en 6 categorías
López et al. (2009)	Universidad de la República, Uruguay	Cualitativa, 31 características revisadas en 11 lenguajes o herramientas

Fuente: presentación propia de los autores.

Estos estudios revisan con detalle las especificaciones de MDA (Object Management Group, 2003a), con el propósito de determinar el conjunto de características que se deben evaluar. En algunos casos los estudios clasifican las características o las herramientas, con el propósito de dar mayor comprensibilidad a su análisis. Los estudios que llevan a cabo una evaluación cualitativa, revisan las características en cada herramienta; mientras que los que realizan una evaluación cuantitativa, definen una escala a partir del soporte que le da la herramienta a las características, para volver discreta la evaluación realizada.

Teniendo en cuenta que los estudios empíricos se realizan mayoritariamente en el marco de MDA, muchas de las características propias de MDSD están

cubiertas por su proximidad con MDA; sin embargo, no sucede lo mismo con los enfoques basados en BPM. Por esto las características se complementan con otras provenientes de dos estudios empíricos adicionales: uno propio de las técnicas de modelado de negocio (Quintero et al., 2008) y otro de técnicas de modelado de arquitecturas de referencia (Quintero y Pérez, 2009). La Tabla 6 ilustra la lista de características deseables en las herramientas que apoyan los enfoques de desarrollo de *software* centrados en modelos, y las referencias en las que se ha trabajado cada característica.

Tabla 6. Características deseables en herramientas para enfoques dirigidos por modelos

#	Característica	Descripción	Principales referencias
P: Características asociadas al proceso			
P1	Definición de transformaciones	Proporciona un lenguaje de modelado para transformaciones y permite manipular las transformaciones	García et al. (2004), Akhter y Tariq (2005) y Quintero y Anaya (2007)
P2	Soporte de transformaciones entre modelos (M2M)	Permite transformaciones de un modelo a otro u otros de menor abstracción	García et al. (2004); Akhter y Tariq (2005) y Mens et al. (2006)
P3	Soporte de transformaciones a código (M2T o M2C)	Permite transformaciones de un modelo a varios tipos de código	Akhter y Tariq (2005) y Quintero y Anaya (2007)
P4	Soporte de ingeniería en reversa	Permite la generación de modelos a partir de código fuente o ejecutables de los sistemas heredados	Akhter y Tariq (2005); Mens et al. (2006) y Quintero y Anaya (2007)
P5	Integración de modelos	Proporciona mecanismos para integrar modelos de un mismo nivel que generan un solo modelo en el siguiente nivel	King's College of London (2003); Wang (2005); García et al. (2004) y Mens et al. (2006)
P6	Consistencia incremental	Conserva los cambios realizados "manualmente" tras regenerar a partir de los modelos	García et al. (2004), Quintero y Anaya (2007)
P7	Trazabilidad	Registra la conexión de los elementos de un modelo con los elementos correspondientes que se generan al transformarlo.	King's College of London (2003); Akhter y Tariq (2005) y Quintero y Anaya (2007)
P8	Definición de roles	Soporta ambientes para el arquitecto, el diseñador, el desarrollador, etc.	Quintero et al. (2008)

Continúa

T: Características asociadas a las técnicas			
T1	Estructura y comportamiento	Es posible definir la parte estática y la parte dinámica del sistema o aplicación en cuestión	Quintero et al. (2008)
T2	Reglas de negocio	Existen técnicas para definir reglas del negocio o restricciones	Quintero et al. (2008), Calic et al. (2008)
T3	Uso de patrones	Tiene técnicas para definir y aplicar patrones para la construcción de modelos y transformaciones	King's College of London (2003), García et al. (2004) y Quintero y Pérez (2009)
T4	Modelado de la GUI	Permite modelos de la interfaz gráfica de usuario o patrones de interacción humano-computador	Calic et al. (2008)
T5	Personalización de la GUI	Dispone plantillas o estilos para la presentación de la aplicación o posibilita su intervención	Calic et al. (2008)
T6	*Framework* de desarrollo	Disponibilidad de un *framework* que permita el desarrollo o extensión de las transformaciones o aplicaciones generadas	Quintero et al. (2008) y Quintero y Pérez (2009)
T7	Entendibilidad	Dispone una terminología comprensible y posee ejemplos que facilitan su utilización	Wang (2005), Quintero y Anaya (2007)
T8	Simplicidad	La transformación es fácil de entender y escribir	Wang (2005)
N: Características asociadas a la notación o al lenguaje			
N1	Soporte de tipos de transformación	Posibilita transformaciones basadas en marcado, metamodelo o patrones	Akhter y Tariq (2005); Quintero y Anaya (2007)
N2	Sintaxis abstracta para el lenguaje de transformación	Define una sintaxis por composición de partes declarativa e imperativa, basada en estándares	King (2003), Wang (2005), Akhter y Tariq (2005) y Mens et al. (2006)
N3	Soporte de perfiles industriales estándar	Permite la adopción de perfiles para plataformas estándar como J2EE, JEE, .NET y PHP	Akhter y Tariq (2005), Quintero y Anaya (2007)
N4	Personalización del lenguaje o notación	Provee definición de perfiles u otros mecanismos para la personalización del lenguaje o notación.	Akhter y Tariq (2005), Quintero y Anaya (2007)

N5	Soporte para la definición de patrones en el código	Patrones en el código con plantillas o mecanismos similares que permitan generación con *idioms* o particularidades en el código	King (2003), García et al. (2004) y Quintero y Pérez (2009)
N6	Metamodelos disponibles	Disponibilidad de los metamodelos utilizados, para posibilitar su extensión o personalización	Quintero et al. (2008) y Quintero y Pérez (2009)
H: Características asociadas a las herramientas			
H1	Robustez	Posibilita dirigir o refinar las transformaciones, y en caso de falla puede continuar transformando	Wang (2005) y Quintero y Anaya (2007)
H2	Interoperabilidad	Es posible importar y exportar modelos usando un estándar de intercambio	King's College of London (2003), Wang (2005) y Mens et al. (2006)
H3	Escalabilidad	Soporta transformación con resultados pequeños o grandes	Wang (2005) Quintero y Anaya (2007)
H4	Soporte para la ejecución	Equipado con herramientas que posibilitan la ejecución de los sistemas generados	García et al. (2004) y Calic et al. (2008)
H5	Pruebas	Equipado con herramientas para la realización de pruebas de ejecución de los modelos, de la aplicación o del código generado	García et al. (2004) y Calic t al. (2008)
H6	Respaldo	Dispone del respaldo de una agremiación reconocida.	Quintero et al. (2008) y Quintero y Pérez (2009)

Fuente: presentación propia de los autores.

Adicional a este conjunto de características, estos estudios empíricos nos dejan otras conclusiones y lecciones aprendidas que resultan de utilidad al momento de pensar en adoptar un enfoque centrado en modelos en el desarrollo de *software*. Según (Quintero y Anaya, 2007), para alcanzar el objetivo de una transformación, se deben utilizar lenguajes que permitan expresar qué se quiere transformar y cómo; sin embargo, existen diversos obstáculos, por ejemplo, que muchas herramientas parten de un modelo de estructura, pues los diagramas de clases son el estándar de facto para representar el PIM (Akhter y Tariq, 2005), lo que limita la expresividad de los modelos de comportamiento en el modelado.

Según (King's College of London, 2003), el potencial de generación es de entre un 50% y un 90%; sin embargo, algunas veces el código generado

es muy pesado o no se corresponde con lo que los desarrolladores esperaban. Adicionalmente, se tiene una curva de aprendizaje grande. Según (García et al., 2004), las posibilidades en cuanto a la transformación cumplen un papel preponderante en las decisiones respecto a la adopción de herramientas; por eso al momento de adoptar un enfoque MDA se hace necesario tener en cuenta factores como la forma de anotar en PIM, relevancia de un PSM, necesidad de ajustar las transformaciones, mecanismos para manipular las transformaciones y si estas son de naturaleza abierta o cerrada.

Para Mens et al. (2006) los dominios de aplicación de la tecnología de transformación de modelos pueden ser muy diversos; por lo tanto, no hay una respuesta única a la pregunta ¿qué enfoque de transformación de modelos es el mejor? Por esto es imprescindible definir cuáles son los criterios de éxito en la adopción de una herramienta de transformación. Todas esta afirmaciones dan mayor validez a una de las conclusiones expuestas en (Wang, 2005) sobre la importancia de unificar los mecanismos de transformación a través de QVT, pues hasta ahora el amplio espectro de alternativas en técnicas, lenguajes y herramientas de transformación constituye un factor de incertidumbre en la adopción de enfoques centrados en modelos; de esta forma, la definición y maduración del estándar QVT se constituye en un paso importante para el éxito de MDA.

5. Conclusiones

Este artículo presenta un amplio conjunto de reflexiones en cuanto a la adopción de enfoques de desarrollo centrado en modelos, que evidencia los pros y los contras encontrados en diversas investigaciones al respecto y que ilustra las principales problemáticas de la transformación de modelos y los trabajos previos en lo que concierne a técnicas, lenguajes y herramientas de transformación de modelos.

A pesar de que los enfoques centrados en modelos se encuentran aún en fase de maduración, las organizaciones ya están empezando a obtener beneficios reales y los fabricantes están demostrando su interés en esta área, por lo que las herramientas son cada vez mejores. Tal situación ha llevado a que los enfoques centrados en modelos estén siendo utilizados en diversos frentes. Vemos, por ejemplo, que la compañía de investigación de mercado Forrester Research Inc., en uno de sus estudios, afirma: "El desarrollo dirigido por modelos (MDD) jugará un papel clave en el futuro del desarrollo de software, MDD es una téc-

nica prometedora para apoyar a los gerentes de desarrollo de aplicaciones frente a la creciente complejidad de los negocios y la demanda" (Lo Giudice, 2007).

En gran parte de los estudios aquí presentados se realiza un profundo análisis de la transformación de modelos, sin poner el relieve en los retos que tiene que enfrentar la MDE. Entre los retos sobresalen los asociados a "la manipulación y gestión de modelos", pues si se realiza un análisis de los principales problemas de MDD, encontramos que la mayoría están relacionados con el manejo de apropiado de los modelos. Mientras que entre los problemas sobresalen los que plantean que en los enfoques centrados en modelos "no se puede describir el tipo de detalles que deben ser implementados", porque de esta forma se imposibilita de generación de aplicación que satisfaga las expectativas de los desarrolladores y, por ende, de los usuarios.

Lo anterior da pie a emprender trabajos que planteen soluciones a una de las principales falencias de los enfoques centrados en modelos, concretada en la siguiente frase: las técnicas, los lenguajes y las herramientas disponibles en la actualidad impiden frecuentemente a los involucrados tener control sobre el proceso y describir completamente el tipo de detalles que la aplicación final debe poseer. Estos detalles se refieren a características como la plataforma, la alineación con los procesos de negocio, los requisitos, las particularidades del código generado, etc.

En el frente de los enfoques basados en BPM, se están realizando grandes esfuerzos tendientes a disminuir la brecha entre un modelo y su representación ejecutable, mediante trabajos que definen las condiciones de mapeo entre BPMN y BPEL, al tiempo que posibilitan la representación directa a entornos de ejecución de procesos; adicionalmente, se tiene la gran ventaja de estar concebido teniendo en cuenta los últimos estándares y aproximaciones basadas en el desarrollo de sistemas de información organizacionales, como la arquitectura orientada a servicios, como se puede evidenciar en (Emig et al., 2006 y 2007).

Aunque en los enfoques basados en BPM también se construye *software* a partir de modelos, los estudios previos tanto en el frente de MDA y MDSD como en el de BPM no han realizado esfuerzo por integrarlos en el marco de la adopción de enfoques centrados en modelos; esto define otro interesante frente de trabajo: realizar un estudio empírico alrededor de la transformación de modelos, que además incluya los enfoques de desarrollo basados en BPM, utilizando como criterios de comparación la lista de características deseables en herramientas para enfoques dirigidos por modelos plateada en este artículo.

Referencias

AKHTER, N. y TARIQ, N. *Comparison of Model Driven Architecture (MDA) based tools*. Estocolmo: Institute of Technology-Karolinska University Hospital, 2005.

ANDROMDA. *10 steps to write a cartridge* [documento en línea]. 2006 <http://andromda.org/index.php?option=com_content&view=category&layout=blog&id=35&Itemid=77> [consulta: 06-07-2010].

BALMELLI, L. et al. Model-driven system development. *IBM System Journal*, 2006, vol. 45, núm. 3, pp. 569-585.

BÉZIVIN, J. In search of a basic principle for model driven engineering. *Upgrade*. 2004, vol. 5, núm. 2, pp. 21-24.

BOOCH, G.; ROMBAUGH, J. y JACOBSON, I. *El proceso unificado de desarrollo de software*. Madrid: Addison Wesley, 1999.

BOOCH, G.; ROMBAUGH, J. y JACOBSON, I. *El lenguaje unificado de modelado*. Madrid: Addison Wesley, 2002.

CALIC, T.; DASCALU, S. y EGBERT, D. Tools for MDA Software Development: Evaluation Criteria and Set of Desirable Features. *5° International Conference on Information Technology: New Generations*, Reno, CSE Department, University of Nevada, 2008.

CHITCHYAN, R. et al. *Survey of analysis and design approaches* [documento en línea]. AOSD, 2005. <http://www.aosd-europe.net/deliverables/d11.pdf> [consulta: 26012010].

CZARNECKI, K. y HELSEN, S. Feature-based survey of model transformation approaches. *IBM System Journal*, 2006, vol. 45, núm. 3, pp. 621-645.

DEN HAAN, J. *8 Reasons Why Model-Driven Approaches (will) Fail* [documento en línea]. 2008. <http://www.infoq.com/articles/8-reasons-why-MDE-fails> [consulta: 06072010].

DE LARA, J. *AToM3 A Tool for Multi-formalism Meta-Modelling* [documento en línea]. 2006. <http://atom3.cs.mcgill.ca/> [consulta: 672010].

DREY, Z. et al. *Kermeta language Reference manual*. s. l.: IRISA, 2010.

DUMAS, M. Case study: BPMN to BPEL Model Transformation. *5th International Workshop on Graph-Based Tools – GraBaTs*. Zurich, 2009.

EMIG, C.; WEISSER, J. y ABECK, S. *Development of SOA-Based Software Systems – an Evolutionary Programming Approach*. s. l.: Universität Karlsruhe, 2006.

EMIG, C. et al. *Model-Driven Development of SOA Services*. s. l.: Universität Karlsruhe, 2007.

FORRESTER CONSULTING. *Modernizing Software Development Through Model-Driven Development. A commissioned study conducted by Forrester Consulting on behalf of Unisys*. s. l., 2008.

FORWARD, A. y LETHBRIDGE, C. Problems and opportunities for model-centric versus code-centric software development: a survey of software professionals. *Proceedings of the 2008 international workshop on Models in software engineering*, Leipzig: *Association for Computing Machinery*, 2008.

FRABCE, R. y RUMPE, B. Model-Driven Development of Complex Software: A Research Roadmap. En: *Future of Software Engineering 2007 at ICSE*. Minneapolis: IEEE, 2007, pp. 37-54.

GAMMA, E. et al. *Design patterns, elements of reusable object-oriented software*. Boston: Addison-Wesley, 1995.

GARCÍA, J. et al. *Un estudio comparativo de dos herramientas MDA: OptimalJ y ArcStyler*. Murcia: Departamento de Informática y Sistemas, Universidad de Murcia, 2004.

GREENFIELD, J. y SHORT, K. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. New York: Wiley, 2004.

GRØNMO, R. y OLDEVIK, J. An empirical study of the UML model transformation tool (UMT). *INTEROP-ESA. First International Conference on Interoperability of Enterprise Software and Applications*, Suiza, 2005.

HAYWOOD, D. *MDA in a Nutshell* [documento en línea]. En: The Server Side, 2004. <http://www.theserverside.com/news/1365166/MDA-Nice-idea-shame-about-the> [consulta: 06-07-2010].

HILL, J. et al. *Magic quadrant for business process management suites* [document en línea]. 2009. <http://mediaproducts.gartner.com/reprints/lombardi/article2/article2.html>.

INFORMATION SOCIETY TECHNOLOGIES (IST). *Modelplex: MODELling solution for comPLEX software systems* [documento en línea]. European Commission, 2006. <http://www.modelplex.org/> [consulta: 09-08-2010].

INTERNATIONAL BUSINESS MACHINES (IBM). *IBM Rational Rose XDE Modeler* [documento en línea]. 2004. <http://www.uml.org.cn/UMLTools/pdf/IB2M.pdf> [consulta: 06-07-2010].

JÉZÉQUEL, J. *Model Transformation Techniques* [documento en línea]. Universidad de Rennes, 2006. <http://www.irisa.fr/prive/jezequel/enseignement/ModelTransfo.pdf> [consulta: 06-07-2010].

KING'S COLLEGE LONDON, UNIVERSITY OF YORK. A*n Evaluation of Compuware OptimalJ Professional Edition as an MDA tool*. York, 2003.

KLEPPE, A.; WARMER, J. y BAST, W. *MDA explained: The practice and promise of model-driven architecture*. New York: Addison-Wesley, 2003.

LARMAN, C. *UML y patrones: introducción al análisis y diseño orientado a objetos*. 4ª ed. Madrid: Prentice Hall, 2005.

LO GIUDICE, D. *The State of Model-Driven Development* [documento en línea], 2007. http://www.forrester.com/rb/Research/clarifying_options_for_application_development_teams/q/id/41357/t/2 [Consulta: 06-07-2010].

LÓPEZ, H. et al. *Estado del arte de lenguajes y herramientas de transformación de modelos*. Montevideo: Instituto de Computación, Universidad de la República, 2009.

MENS, T. y van GORP, P. A taxonomy of model transformation and its application to graph transformation. *Electronic Notes in Theoretical Computer Science*. 2006, núm. 152, pp. 125-142.

MENS, T. et al. Applying a model transformation taxonomy to graph transformation technology. *Electronic Notes in Theoretical Computer Science*. 2006, núm. 152, pp. 143-159.

MOHAGHEGHI, P. et al. *MDE adoption in industry: challenges and success criteria*. New York: Springer, 2009.

MOHAGHEGHI, P. y AAGEDAL, J. Evaluating quality in model-driven engineering. *Proceedings of the International Workshop on Modeling in Software Engineering 2007*. International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 2007.

NORTHROP, L. *Software product line essentials*. s. l.: Software Engineering Institute, Carnegie Mellon University, 2008.

OASIS. *Web Services Business Process Execution Language Version 2.0* [documento en línea]. 2007. <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf> [consulta: 06-06-2010].

OBJECT MANAGEMENT GROUP. *Model Driven Architecture (MDA) Guide v1.0.1* [documento en línea]. 2003a. <http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf> [consulta: 06-07-2010].

OBJECT MANAGEMENT GROUP. *Common Warehouse Metamodel (CWM) Specification v1.0.1* [documento en línea]. 2003b. <http://www.omg.org/spec/CWM/1.1/PDF/> [consulta: 06-97-2010].

OBJECT MANAGEMENT GROUP. *Meta Object Facility (MOF) Core Specification v2.0*. [documento en línea]. 2006. <http://www.omg.org/spec/MOF/2.0/PDF/> [consulta: 06-07-2010].

OBJECT MANAGEMENT GROUP. *MOF 2.0 / XMI Mapping, v2.1.1* [documento en línea]. 2007a. <http://www.omg.org/spec/XMI/2.1.1/PDF/index.htm> [consulta: 06072010].

OBJECT MANAGEMENT GROUP. *Object Constraint Language v2.2* [documento en línea]. 2007b. <http://www.omg.org/spec/OCL/2.2/PDF> [consulta: 06072010].

OBJECT MANAGEMENT GROUP. *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification v1.1* [documento en línea]. 2009a. <http://www.omg.org/spec/QVT/1.1/Beta2/PDF> [consulta: 06072010].

OBJECT MANAGEMENT GROUP. *Business Process Modeling Notation (BPMN) FTF Beta 1 v2.0* [documento en línea]. 2009b. <http://www.omg.org/spec/BPMN/2.0/Beta1/PDF/> [consulta: 06072010].

OBJECT MANAGEMENT GROUP. *OMG Unified Modeling Language TM (OMG UML), Superstructure v2.3* [documento en línea]. 2010. <http://www.omg.org/spec/UML/2.3/Superstructure/PDF/> [consulta: 06072010].

QUINTERO, J. y ANAYA, R. Marco de referencia para la evaluación de herramientas basadas en MDA. *Memorias del X Workshop IDEAS*, 2007. p. 225-238.

QUINTERO, J.; CADAVID, J. y OSPINA, C. Estudio comparativo de técnicas de modelado de negocio. En: *Memorias del XI Workshop IDEAS*, 2008, pp. 309-314.

QUINTERO, J. y PÉREZ, J. Estrategias para la definición de una técnica de modelado para arquitecturas de referencia. *Memorias del XII Workshop IDEAS*, 2009.

SELIC, B. The pragmatics of model-driven development. *IEEE Software*, vol. 5, núm. 20, pp. 10-25.

SENDALL, S. y KOZACZYNSKI, W. *Model Transformation - the Heart and Soul of Model-Driven Software Development*. Geneva: Software Modeling and Verification Lab., University of Geneva, 2003.

SOMMERVILLE, I. *Ingeniería de software*. 7ª ed. Madrid: Pearson Addison Wesley, 2005.

SUN DEVELOPER NETWORK. *Java Metadata Interface (JMI)* [documento en línea]. 2002. <http://java.sun.com/products/jmi/> [consulta: 06072010].

THE ECLIPSE FOUNDATION. *ATL User Guide* [documento en línea]. 2006. <http://wiki.eclipse.org/ATL/User_Guide> [consulta: 06072010].

VÖLTER, M. y STAHL, T. *Model-Driven Software Development (Technology, Engineering, Management)*. New York: Wiley, 2006.

WANG, W. *Evaluation of UML Model Transformation Tools*. Viena: Business Informatics Group, Vienna University of Technology, 2005.

W3C. *XSL Transformations (XSLT). v1.0* [documento en línea]. 1999. <http://www.w3.org/TR/xslt> [consulta: 06072010].