Una aproximación a la generación automática de código en un contexto MDD sobre modelos BPMN

Ignacio Martínez A., Lautaro Mendez, Julián Perelli, Mauricio Pérsico, Nahuel Santos B.

Cátedra de Ingeniería de Software, Departamento de Sistemas de Información
Universidad Tecnológica Nacional, Facultad Regional La Plata

{imartinez,lmendez,jperelli,mpersico,nsantos}@linsi.edu.ar

Resumen. Este trabajo expone los esfuerzos realizados para lograr una transformación entre modelos, en un contexto de desarrollo dirigido por modelos (MDD), mediante la elaboración de un prototipo, partiendo desde la especificación del metamodelo del lenguaje BPMN, UML (diagramas de actividades) y las reglas de transformación pertinentes. El prototipo se implementó como un plugin para la plataforma de desarrollo Eclipse. El resultado obtenido es un prototipo de herramienta visual que provee al desarrollador los recursos necesarios para poder definir un metamodelo origen, un metamodelo destino y las correspondientes reglas de transformación entre uno y otro, las cuales serán aplicadas para la generación automática de un modelo final a partir de otro inicial. Gracias al enfoque dirigido por modelos que pretende cubrir este plugin, se favorece ampliamente la fabricación de software, brindando al encargado del desarrollo más tiempo para preocuparse por cuestiones relacionadas al diseño y la captura fiable de requisitos estrechamente relacionadas con el dominio, en detrimento del tiempo que debería utilizar en cuestiones más tecnológicas que no afectan directamente con la resolución del problema.

Palabras clave: MDD, BPMN, Eclipse Modeling Framework (EMF), Graphical Modeling Framework (GMF), QVT (Query/View/Transformation).

1 Introducción

Este trabajo describe un proceso de desarrollo dirigido por modelos [1], [2], tomando como base un lenguaje estándar para procesos de negocios llamado BPMN [3], [4]. Para la generación de este proceso, se construyó una herramienta (prototipo) a través de Eclipse RCP [5], [6], [7], utilizando el Graphical Modeling Framework [8] de Eclipse, permitiendo a los desarrolladores de software generar modelos BPMN de manera gráfica. Para esto se construyó un metamodelo simplificado de lenguaje BPMN, sobre el cual se basa el modelo creado por el usuario de la herramienta, junto con algunas transformaciones básicas de modelo a modelo. Esto permite al usuario de la herramienta centrarse en el diseño de un modelo consistente con las especificaciones del negocio, y no concentrarse en buscar soluciones a problemas que

no son específicos del dominio de la aplicación [9], [10]. En el presente trabajo se muestra una posible estrategia desde la óptica de MDD [11], a través de la transformación de modelos BPMN a modelos UML [12]. Utilizar los frameworks disponibles públicamente como GMF y QVT [13] de Eclipse, permite la fácil ampliación, modificación y configuración de las funcionalidades de la herramienta, como por ejemplo, realizar transformaciones de modelo a modelo, o de modelo a texto, según se requiera.
En la sección 2 del documento se describen los metamodelos y modelos utilizados en el presente trabajo. En la sección 3 se explica el proceso de desarrollo del plugin, en conjunto con los pasos necesarios para la utilización de un metamodelo de entrada, como es el caso de BPMN. En la sección 4 se procede a definir las transformaciones de modelo a modelo utilizada, en conjunto con una breve reseña sobre el estándar QVT. Finalmente, en la sección 5 se presentan los pasos finales en el desarrollo del plugin y de la puesta en marcha de la aplicación RCP.

2 Modelos, metamodelos y transformaciones

Para realizar una transformación entre modelos, como la representada en la Fig. 1, se debe contar con información de los metamodelos que describen las representaciones de cada elemento del modelo. De esta manera, una transformación es una correlación entre un modelo representado por un metamodelo, a un segundo modelo, representado por un segundo metamodelo.

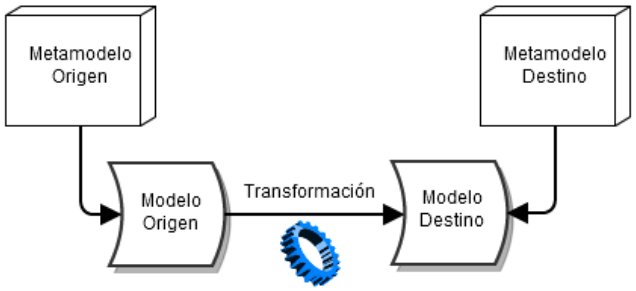

Fig. 1: Descripción del proceso de transformación entre modelos

Cada metamodelo define las restricciones que debe cumplir un modelo. De esta manera, siempre que las restricciones se cumplan, puede definirse una regla de transformación que haga que los modelos sean consistentes el uno con el otro. La herramienta desarrollada fue realizada utilizando Eclipse RCP. Se estructuró creando un plugin principal (BPMN), generado con GMF, el cual define las herramientas para visualización y edición de diagramas BPMN. El plugin QVT brinda las dependencias necesarias para la traducción y un lenguaje para el mismo fin. El plugin de GMF brinda el soporte a la edición gráfica de modelos Ecore. En la Fig. 2 se muestra un esquema conceptual de las tecnologías empleadas para este proyecto.

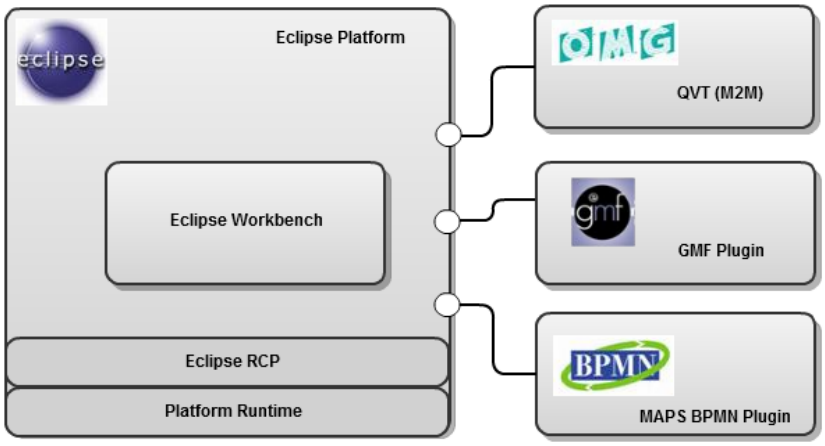

Fig. 2: Componentes del proyecto realizado

El metamodelado es una herramienta necesaria para la buena formación de modelos que carecen de ambigüedad y falta de precisión, como así también para establecer reglas de transformación usadas en MDD con las cuales se pueden obtener modelos que son diferentes, pero equivalentes en cuanto a lo que representan.

2.1 Metamodelo BPMN

BPMN es un lenguaje notacional estándar - fuertemente visual - soportado por la OMG. Fue específicamente diseñado para representar gráficamente una secuencia de actividades que ocurre en un proceso de negocio. Permite abstraer muchos detalles que no están involucrados directamente con procesos de negocios y centrarse directamente en este tipo específico de aplicaciones. BPMN fue creado específicamente para que puedan utilizarlo personas que sean parte de la gerencia de una organización, que entienden concretamente los procesos de negocio. Mientras que BPMN2 [14], [15] es implementando como un perfil de UML [16], el cual extiende el metamodelo de UML, incluyendo restricciones, modificaciones y extensiones la semántica de UML para asegurar la equivalencias semánticas con BPMN. Su capacidad de modelado y complejidad resulta mucha más simple, lo que permite poder generar transformaciones de manera más fácil, ya sea a otros modelos (que podrían ser UML) como a texto (código JAVA [17] por ejemplo). Para la transformación de modelos expresados en BPMN, se define de manera formal su sintaxis a través de un metamodelo, representando una instancia del estándar MOF [18].

2.2 Modelado con UML

UML es un estándar de facto para el modelado de propósito general, mantenido y desarrollado por la OMG [21]. En UML existen diagramas de actividad que representan la dinámica (o comportamiento) de un modelo de sistema. Estos diagramas se utilizan para describir las actividades, decisiones, interacciones y la concurrencia de un flujo de trabajo o un proceso [22]. BPMN y los diagramas de actividad de UML tienen el mismo propósito y comparten muchas características

similares. Ambos lenguajes de modelado se utilizan para flujos de trabajo de modelos o procesos de negocio y la notación para cada lenguaje puede ser traducido a otro lenguaje bajo alguna condición. Sin embargo, la decisión sobre que lenguaje de modelado se debe utilizar es bastante difícil para el diseñador. Hay limitaciones, tanto para BPMN como en los diagrama de actividad de UML. La utilización de UML permite al equipo de desarrollo y de diseño, una visión más general sobre los requerimientos.
En el presente proyecto se utiliza la especificación de UML desarrollada por Eclipse como metamodelo destino, y se utilizan determinadas reglas de transformación entre cada uno de los modelos [23]. Cabe señalar que en el momento de creación de este trabajo, no existe soporte desde OMG para proveer un metamodelo de UML 2.0 debido a problemas estructurales declarados por la organización.
Esta transformación de BPMN a diagramas de actividades genera una mayor abstracción y simpleza, más cercana a la visión del equipo de desarrollo.

3 Creación del plugin con GMF

El proceso de creación de un editor gráfico a través de GMF posee 6 etapas cuyo producto final está representado por un plugin de Eclipse en estado puro, o en su defecto una aplicación Eclipse RCP. Las etapas pueden resumirse de la siguiente forma:

- **a.** Creación del metamodelo: utilizado para la creación de nuevos modelos de parte del usuario.
- **b.** Creación del set de herramientas del plugin: proporciona la forma de uso del plugin final, a través de los elementos gráficos disponibles al usuario.
- **c.** Creación del modelo de mapeo gráficos: necesario para el mapeo de nuevos modelos definidos por el usuario a través de las herramientas gráficas.

En la Fig. 3 se muestra la interconexión de las distintas tecnologías, generando productos intermedios, durante la creación de un proyecto GMF.
En los siguientes apartados se explicará cada una de las etapas.

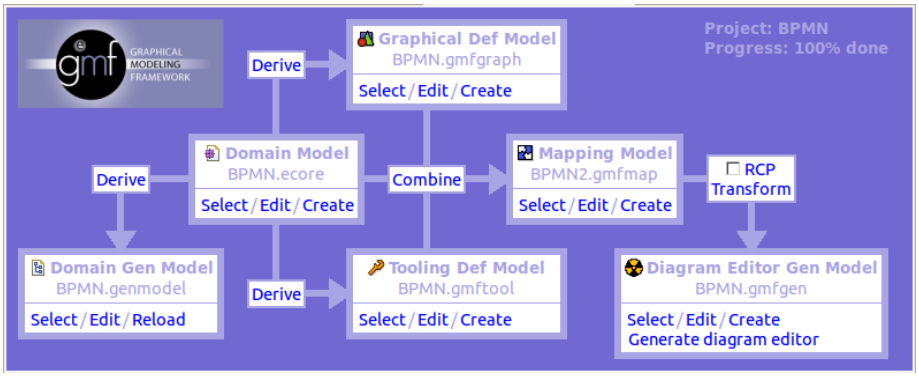

Fig. 3: Dashboard del desarrollo de un proyecto GMF

3.1 Definición del metamodelo (Ecore metamodel)

EMF [24], [25] está basado en 2 metamodelos [26]: el Ecore y el Genmodel. El metamodelo Ecore contiene la información sobre las clases definidas. El modelo Ecore permite definir diferentes elementos:

- **a.** EClass: representa un atributo que contiene el nombre y el tipo.
- **b.** EAtribute: representa un atributo que tiene un nombre y un tipo.
- **c.** EReference: representa un extremo de la asociación entre dos clases. Tiene un flag para indicar si es un contenedor y una referencia a la clase que apunta.
- **d.** EDataType: representa el tipo de un atributo.

El modelo Ecore posee una estructura de árbol. Dentro de este modelo existe un objeto raíz que representa al modelo Ecore. Este modelo tiene hijos que representan paquetes, y sus hijos representan las clases, mientras que los hijos de las clases representan los atributos de estas clases. El otro metamodelo utilizado GenModel, contiene información adicional para la generación de código, por ejemplo el directorio y la información del archivo. Produce las clases de implementación java para un modelo o instancia del metamodelo Ecore. El GenModel también contiene los parámetros de control sobre cómo debe ser generado el código. Define los mapeos necesarios entre el metamodelo y definiciones para configurar dos plugins llamados edit y editor que contienen operaciones de bajo nivel que necesita el plugin que se generará para funcionar.

3.2 Definición gráfica del metamodelo (GMFGraph)

GMF permite generar la infraestructura y componentes necesarios en tiempo de ejecución para desarrollar editores gráficos. Al estar basado en Ecore EMF y GEF, es utilizado para producir un editor gráfico que genera instancias de metamodelos [27]. GMFGraph es utilizado para definir los elementos gráficos de nuestro dominio. En este caso, vinculados con cada uno de los elementos del modelo de dominio o metamodelo Ecore BPMN.

3.3 Configuración de las barras de herramientas (GMFTool)

Este archivo es utilizado para definir la paleta de herramientas que puede usarse en el editor gráfico**.** En cada una de las barras de herramientas definidas, se podrá hacer uso de los elementos que deban estar disponibles al usuario. Por ejemplo, en el metamodelo BPMN definido existen ciertas clases como es el caso de ProcessObject, las cuales no deben poder ser insertadas en el modelo creado por el usuario, de esta forma, al momento de crear las barras de herramientas con los elementos disponibles, se deberán considerar cuales de los mismos son necesarios y cuales no.

3.4 Mapeo de configuraciones con el modelo GMFMap

En este archivo se vinculan o mapean los modelos originados por las anteriores configuraciones: el modelo de dominio Ecore, el modelo gráfico GMFGraph y el modelo de herramientas GMFTool. GMFMap es quien permite la generación de código del editor. Como se observa en la Fig. 4, la forma de identificar visualmente los objetos de la clase Process es a través de rectángulos redondeados con borde

sólido. Así, cada una de las representaciones gráficas del metamodelo utilizado se encontraran en el archivo .gmfmap.

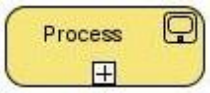

Fig. 4: Notación gráfica de un elemento de tipo Process

3.5 Generación del plugin a través de GMFGen

Este archivo contiene la información necesaria para poder generar el plugin que hará posible la edición gráfica de modelos definidos a través del metamodelo desarrollado. GMFGen se compila mediante GMF, obteniendo como producto final el plugin desarrollado. De esta forma se logra un editor gráfico en base a GMF y la utilización de EMF (GenModel).

4 Definición de las transformaciones a aplicar M2M

En MDD, hay lenguajes de transformación de modelos, que toman modelos de entrada conformes a un metamodelo, y produce una salida conforme a otro metamodelo, el cual puede o no ser el mismo que el metamodelo de entrada, como por ejemplo la transformación de BPMN a UML, o cuando se realiza refactoring sobre el mismo modelo. Ejemplos de estos lenguajes son: ATL [28], MOFM2T [29], QVT, TXL [30]. Eclipse M2M [31] es una implementación del estandar OMG QVT. Dentro de esta se encuentra ATL.

4.1 QVT (Query/View/Transformation)

El estándar QVT son un conjunto de lenguajes para la modelización de transformaciones, las cuales son las bases de MDD. Este estándar es otro de los que se encuentra administrado por OMG. En la Fig. 5 se observa un ejemplo de transformación especificado en el lenguaje QVT. Como se puede apreciar, en base a los metamodelos de origen y destino, BPMN y UML respectivamente, se especifican las reglas de transformación a través de la sintaxis de QVTO [32] (Operational QVT Language).

```
modeltype BPMN uses "bpmn";
modeltype UML uses "http://www.eclipse.org/uml2/2.0.0/UML";

transformation Ecore2UML(
    in ecore: BPMN,
    out uml: UML
);

main() {
    ecore.objects()[Pool]->map toActivityPartition();
    ecore.objects()[Task]->map toActivity();
    ecore.objects()[Subprocess]->map toStructureActivityNode();
    ecore.objects()[StartEvent]->map toInitialNode();
    ecore.objects()[EndEvent]->map toFinalNode();
}

mapping Pool::toActivityPartition() : UML::ActivityPartition {
    name := self.name;
}
```

Fig. 5: Ejemplo de transformación entre BPMN y UML.

5 Despliegue del prototipo

Para la generación del plugin BPMN (automático), con la ayuda de GMF, una vez que se realizaron los pasos de configuración definidos en la parte 3.3, sólo hace falta realizar click con el botón derecho del mouse en el archivo GMFGen y seleccionar "Generar Plugin". De esta manera, las herramientas de GMF se ocupan de generar automáticamente todo el código necesario para el funcionamiento del plugin de edición gráfico.

La configuracion de la app RCP, splash screen e iconos se realiza a través de la edición de los archivos .MANIFEST y .product que son parte del entorno de desarrollo de Eclipse. RCP permite definir, entre otras cosas, cuales son los plugins a agregar, en qué orden deben ser ejecutados, una pantalla gráfica de introducción (splash screen), íconos del ejecutable. Y finalmente se genera un solo archivo ejecutable a partir de la aplicación RCP (app standalone - automático). La aplicación terminada es una herramienta amigable que se le presenta al usuario muy simple de utilizar. De esta manera, es una herramienta portable, que puede ser ejecutada en distintas plataformas (windows, linux, macOS) que tengan soporte para JavaVM.

6 Conclusiones

Todo el entorno de desarrollo Eclipse con su gran cantidad de plugins que amplían su potencial sumado al hecho que es de código abierto, hacen que sea una herramienta con gran potencial y capacidad para lograr trabajos de excelente calidad. Sin embargo, por ser un proyecto tan vasto se puede notar cierta dificultad para poder adquirir los conocimientos y habilidades necesarias para realizar un uso efectivo del mismo. Existe un gran soporte de parte de la comunidad de desarrolladores, que sumado a la certeza de su potencial, nos permite poder explotar esta gran herramienta con el consecuente esfuerzo y tiempo necesarios. Durante este desarrollo se logró una buena integración de varias tecnologías que nos provee Eclipse, permitiendo crear herramientas de transformación de modelos de muy alto nivel sin mayores inconvenientes.

Se concluyó obteniendo un prototipo inicial que permite el diseño a través de BPMN y de transformaciones básicas de algunos elementos BPMN a correspondencias en un diagrama de actividad de UML2.0. Aunque aún falta camino que recorrer para lograr una transformación que nos brinde código en un determinado lenguaje, es decir, modelo a texto, se pudo culminar un paso importante que nos muestra el camino a seguir para poder completar un proceso de desarrollo dirigido por modelos haciendo uso de herramientas de código abierto. Las herramientas de Eclipse presentan cierta dificultad en su aprendizaje, pero se puede realizar una buena integración entre ellas, permitiendo crear nuevas herramientas de transformación de modelos de alto nivel de abstracción, independientes de la plataforma tecnológica.

7 Trabajos futuros

Se ha delineado el camino a seguir y se construyeron las bases necesarias para poder seguir en esta misma línea y así poder lograr expandir las reglas de transformación contempladas para el prototipo. Esto último no sólo posibilita ampliar en detalle y

cantidad los modelos destinos logrados con las transformaciones, sino que nos acerca al objetivo de generar código a partir de modelos. Se prevé el uso de OCL [33] como mecanismo de validación para los modelos BPMN y su metamodelo de transformación. Para lograr todo esto es fundamental tener un metamodelo UML estable y estandarizado por OMG. La implementación de estas reglas se hizo utilizando especificaciones de terceros correspondientes a los diagramas de actividades. Se espera que en el corto plazo se pueda contar con una especificación documentada del metamodelo de UML 2.0.
También se pretende, concretar transformaciones a otros lenguajes como BPEL, y luego a código java, para cerrar un ciclo completo dirigido por modelos. Lograr esto, permitirá combinar la herramienta con las emergentes metodologías ágiles, para mostrar su importancia en el marco del del proceso de desarrollo de software.

Referencias

[1] Scott W. Ambler.“The object primer: agile modeling-driven development with UML 2.0”. 3ra edición. Cambridge University Press, pp. 101-133 (2004). ISBN-10: 0521540186
[2] Ian Sommerville. “Ingeniería del Software”. Séptima edición. Pearson Education S.A., pp. 11, 79-83 (2005). ISBN-10: 8478290745.
[3] OMG. Business Process Modeling Notation (BPMN) - Version 2.0 (2011). http://www.omg.org/spec/BPMN/2.0/
[4] Weske Mathias, “Business Process Management: Concepts, Languages, Architectures”. Springer (2010). Pag 3-67. ISBN-10: 3642092640
[5] The Eclipse Foundation open source community website. http://www.eclipse.org
[6] McAffer, Jean-Michel LeMieux y Chris Aniszczyk. "Eclipse Rich Client Platform". Segunda Edición. Addison-Wesley Professional (2010). ISBN-10: 0321612345
[7] Vladimir Silva. "Practical Eclipse Rich Client Platform Projects". Springer (2009). ISBN-10: 1430218274
[8] Graphical Modeling Framework (GMF) Project. http://www.eclipse.org/gmf/
[9] Craig Larman. “Agile and Iterative Development: a manager's guide”. Pearson Education, Inc. Addison-Wesley Professional, pp. 9-20, 25-35 (2004). Agile software development series. ISBN-10: 0131111558.
[10] Thomas Stober y Uwe Hansmann. “Agile Software Development: best practices for large software development projects”. Springer-Verlag Berlín Heidelberg, pp. 37-57 (2009). ISBN-10: 3540708308.
[11] Pons Claudia, Giandini Roxana y Pérez Gabriela. “Desarrollo de Software Dirigido por Modelos: conceptos teóricos y su aplicación práctica”. 1er. edición. EDULP & McGraw-Hill Educación, Argentina, pp. 28-38 (2010). ISBN-13: 9789503406304
[12] OMG. Unified Modeling Language (UML) - Version 2.4.1 (2011). http://www.omg.org/spec/UML/2.4.1/
[13] OMG. Query/View/Transformation (QVT). Version 1.1 (2011). http://www.omg.org/spec/QVT/1.1/
[14] OMG. Business Process Modeling Notation (BPMN) - Version 2.0 (2011). http://www.omg.org/spec/BPMN/2.0/
[15] Bruce Silver. "BPMN Method and Style: A Levels-based Methodology for BPM Process Modeling and Improvement Using BPMN 2.0". Cody-Cassidy Press (2009). ISBN-10: 0982368100
[16] The Eclipse Foundation. "Introduction to UML2 Profiles - Eclipse". http://www.eclipse.org/modeling/mdt/uml2/docs/articles/Introduction_to_UML2_Profiles/article.htm

[17] Lenguaje JAVA. http://www.java.com
[18] OMG. Meta Object Facility (MOF) Core Specification.Versión 2.4.1 (2011). http://www.omg.org/spec/MOF/2.4.1/
[19] Roxana Giandini, Gabriela Pérez, Claudia Pons. "Un lenguaje de Transformación específico para Modelos de Proceso del Negocio". XXXVIº Conferencia Latinoamericana de Informática (CLEI 2010). Asunción, Paraguay
[20] Metamodelo BPMN que propone Eclipse. Tutorial BPMN (2006). http://wiki.eclipse.org/Graphical_Modeling_Framework/Tutorial/BPMN
[21] Object Management Group (OMG). www.omg.org
[22] Grady Booch, Ivar Jacobson y James Rumbaugh. "El lenguaje unificado de modelado". Segunda Edición. Pearson Education, S.A., pp. 3-40 (2006). ISBN-13: 9788478290765.
[23] Nguyễn Quốc Bảo. "A proposal for a method to translate BPMN model into UML activity diagram". 13th International Conference on Business Information Systems. 2010.
[24] Eclipse Modeling Framework (EMF) Project. http://www.eclipse.org/modeling/emf/
[25] Dave Steinberg, Frank Budinsky, Marcelo Paternostro y Ed Merks."EMF: Eclipse Modeling Framework". Segunda Edición (2009). Addison-Wesley Professional. ISBN-10: 0-321-33188-5
[26] Tutorial de EMF. http://www.vogella.com/articles/EclipseEMF/article.html
[27] Tutorial de GMF. http://www.kermeta.org/docs/fr.irisa.triskell.kermeta.samples.fsm.documentation/build/html.chunked/KerMeta-Create-FSM-Graphical-Editor-With-GMF/ch02.html
[28] Bézivin, J, Dupé, G, Jouault, F, Pitette, G, and Rougui, JE. "First experiments with the ATL model transformation language: Transforming XSLT into XQuery". OOPSLA 2003 Workshop, Anaheim, California. 2003.
[29] MOF Model to Text Transformation Language (MOFM2T). http://www.omg.org/spec/MOFM2T/1.0/
[30] QVT·TXL. J.R. Cordy, "Excerpts from the TXL Cookbook", Generative and Transformational Techniques in Software Engineering, LNCS 6491, January 2011, pp. 27-91.
[31] Model 2 Model (M2M). http://www.eclipse.org/m2m/
[32] Operational QVT (QVTO). http://www.eclipse.org/m2m/qvto/doc/M2M-QVTO.pdf
[33] Jos Warmer y Anneke Kleppe. "The Object Constraint Language: getting Your Models Ready for MDA". Pearson Education, pp. 3-18 (2003), ISBN-10: 0321179366.