Plan de Trabajo Final
Carrera Ingeniería de Sistemas
Facultad de Ciencias Exactas-UNICEN

Tema: Desarrollo de herramientas de metamodelado formal

Alumno: Daniel Duarte

Directora: Liliana Favre

1. INTRODUCCIÓN

El desarrollo de software dirigido por modelos (Model-Driven Development, MDD) emergió en la última década como un nuevo camino hacia la industrialización de la producción de software. La realización específica de MDD propuesta por OMG (Object Management Group) es MDA (Model Driven Architecture) [10] [11]. MDA propone separar la especificación de la funcionalidad del sistema de su implementación sobre una plataforma y tecnología específica y controlar la evolución del software desde modelos abstractos a implementaciones tendiendo a aumentar el grado de automatización.

Tres conceptos fundamentales están asociados con MDA: modelos, metamodelos y transformaciones. En particular, MDA desplazó el centro de atención de modelos a metamodelos que permiten describir tanto familias de modelos como de transformaciones entre modelos. Este rol central de los metamodelos requiere de la existencia de técnicas específicas para su evolución. El estándar propuesto por OMG para metamodelado es el meta-metamodelo MOF (Meta Object Facility)[12]. Todos los estándares MDA son definidos como metamodelos MOF. Las herramientas CASE basadas en MDA soportan niveles de meta-metamodelado MOF que permiten a un usuario definir sus propios lenguajes de modelamiento [5].

MOF tiene una semántica semiformal expresada a partir de diagramas de clase UML, restricciones OCL (Object Constraint Language) [13] y lenguaje natural. Sin embargo, sería ventajoso integrarlo con lenguajes formales que permitan realizar validaciones y verificaciones de modelos a partir de los sólidos resultados logrados por la comunidad de métodos formales [2].

En este Trabajo Final de la carrera de Ingeniería de Sistemas se propone desarrollar herramientas que soporten metamodelado formal, a partir del formalismo algebraico.

A continuación se describen los conceptos básicos del enfoque que propone OMG para desarrollo dirigido por modelos, la arquitectura dirigida por modelos MDA, y las características de MOF.

▪ *Model Driven Architecture (MDA)*

La definición inicial de MDA tuvo que ver con el problema de interoperabilidad de middleware en Internet. Más allá de resolver problemas de interoperabilidad, MDA ofrece otros beneficios vinculados a productividad, calidad de procesos y costos de mantenimiento. En un proceso dirigido por modelos basado en MDA pueden distinguirse diferentes tipos de modelos:

- CIM (*Computation Independent Model*), que describe a un sistema desde un punto de vista independiente de la computación, este modelo usualmente es llamado modelo de negocio o modelo del domino del sistema.

- PIM (*Platform Independent Model*), que expresa al sistema desde un punto de vista independiente de la plataforma.

- PSM (Platform Specific Model), relacionado a una plataforma específica (por ejemplo .NET, JEE, relacional, web, móvil, etc.)

- Modelos de implementación o código que algunos autores denominan ISM (*Implementation Specific Model*).

La difusión inicial de MDA enfatizaba en su relación con UML como lenguaje de modelado, sin embargo hay usuarios de UML (*Unified Modeling Language*) que no usan MDA y usuarios de MDA que usan otros lenguajes de modelado como específicos DSL (*Domain Specific Language*) [4].

Una de las características esenciales de MDA es que todos los artefactos involucrados en un proceso de desarrollo se representan a partir del lenguaje de metamodelado MOF. MOF define una forma común de capturar todas las construcciones de modelado e intercambio que son usadas en MDA y es la esencia de MDA al permitir que diferentes tipos de artefactos provenientes de diferentes vendedores sean usados juntos en un mismo proyecto.

Los conceptos de modelamiento en MOF son: clases, que modelan MOF meta-objetos; asociaciones que modelan relaciones binarias entre meta-objetos; tipos de datos y paquetes para modularizar modelos. OCL es utilizado para adjuntar restricciones a los componentes del metamodelo. MOF se autodescribe, es decir, se define usando sus propias construcciones de modelamiento. Esto provee un tratamiento uniforme de la semántica entre artefactos que representan modelos y metamodelos.

Otro concepto fundamental es el de transformaciones entre modelos. En MDA, una transformación es la especificación de mecanismos para convertir elementos de un modelo en elementos de otro modelo. El estándar propuesto por OMG para especificar transformaciones es el metamodelo QVT (*Query, View, Transformation*) [14]. QVT define tres lenguajes de transformación de modelos (QVT-Operational, QVT-Relational y QVT-Core) que conforman a MOF 2.0. Una transformación en cualquiera de estos lenguajes puede considerarse como un modelo que conforma a uno de los metamodelos especificados en el estándar. QVT está integrado con el estándar OCL 2.0 [13].

Algunos de los subproyectos del proyecto Eclipse [7] proporcionan herramientas y entornos de ejecución alineados con estándares de MDA. Por ejemplo, EMF (*Eclipse Modeling Framework*) provee facilidades para metamodelado y un motor de ejecución para los modelos que soporta la creación de clases Java de un modelo. En particular, provee a Ecore, una implementación parcial de MOF en Eclipse y un metamodelo Ecore de UML2.0 [16]. También, existen subproyectos relacionados con lenguajes de transformaciones de modelos como QVT aún en estado de incubación.

2. MOTIVACIÓN

Los metamodelos cubren un rol central en desarrollos dirigidos por modelos. Tal como fue expresado anteriormente el estándar de OMG para metamodelado es MOF, un meta-metamodelo que define una manera común de capturar la diversidad de estándares de modelamiento y construcciones de intercambio usadas en MDA. La sintaxis abstracta de MOF se define a partir de diagramas de clase UML especificados mediante restricciones OCL y lenguaje natural.

Un problema central en MDA es definir metamodelos "correctos" y alineados con MOF dado que las inconsistencias en la especificación de los metamodelos afectará a los modelos y sus implementaciones.

La integración de especificaciones MOF con especificaciones formales puede ser un complemento beneficioso para el análisis de consistencia de metamodelos. Una especificación formal permite construir especificaciones de software precisas y analizables, validar transformaciones entre modelos y ser un punto de partida para implementaciones.

En esta dirección, han surgido propuestas, una de las más recientes es FORMULA, un framework , basado en el formalismo lógico, para especificar DSL y transformaciones de modelos [9]. En [8] se presenta NEREUS, un lenguaje de metamodelado formal. NEREUS puede ser considerado como una notación intermedia que puede ser integrada con lenguajes algebraicos como por ejemplo CASL (Common Algebraic Specification Language) [3]. En [8] se muestra cómo pueden automatizarse transformaciones de metamodelos MOF en NEREUS y cómo puede integrarse NEREUS con otros lenguajes algebraicos. La integración de MOF y NEREUS puede automatizarse, lo cual es fundamental por razones de escalabilidad. En cierta forma podríamos decir que así como MOF es un DSL para definir metamodelos semiformales (expresados en UML y OCL), NEREUS puede ser visto como un DSL para definir metamodelos formales. Otra ventaja de NEREUS está vinculada a razones pragmáticas: NEREUS es un lenguaje formal con una sintaxis concreta cercana a MOF y que por ende facilita a los metadiseñadores entender las especificaciones formales.

Tal como MOF, NEREUS provee construcciones sintácticas para clases, asociaciones binarias, tipos de datos y paquetes. Podríamos decir que NEREUS es una posible sintaxis concreta para MOF, extendida con la especificación de propiedades adicionales expresadas por axiomas.

Se considera de interés contar con herramientas que soporten metamodelado formal y que aporten a la comunidad MDA. Por este motivo se propone desarrollar herramientas de soporte para NEREUS. A continuación se precisa el objetivo y alcance de la propuesta.

3. OBJETIVO Y ALCANCE DEL TRABAJO FINAL

El objetivo de este Trabajo Final de la carrera de Ingeniería de Sistemas es desarrollar herramientas para la evolución de metamodelos en el contexto de desarrollos basados en MDA. Se propone integrar especificaciones semiformales de metamodelado con especificaciones algebraicas, específicamente, desarrollar herramientas de soporte para metamodelado formal que permitan editar metamodelos MOF en una sintaxis concreta

textual cercana a MOF y con axiomas al estilo de los lenguajes de especificación algebraica. El formalismo algebraico permite realizar validaciones de estas especificaciones en un marco riguroso.

Se propone usar al lenguaje NEREUS para la especificación de metamodelos. A partir de la definición del lenguaje NEREUS, se especificará una EBNF (Extended Backus Naur Form) de NEREUS y se desarrollará un parser del lenguaje NEREUS usando el generador de parser ANTLR (ANother Tool for Language Recognition). ANTLR toma como entrada una EBNF de NEREUS y genera el código fuente de un reconocedor de especificaciones NEREUS. Esta es la etapa central del desarrollo [1] [15].

Teniendo en cuenta que la semántica del lenguaje NEREUS fue dada por traducción al lenguaje CASL, se propone desarrollar un traductor de NEREUS a CASL a partir de los resultados presentados en [8] y las herramientas provistas por COFI (The Common Framework Initiative for Algebraic Specification and Development) [6]. La propuesta se validará mediante el desarrollo de casos de estudio asociados a metamodelos simplificados.

4. CRONOGRAMA DE ACTIVIDADES

Se proponen las siguientes etapas para concretar este plan:

1. Estado actual del conocimiento y búsqueda bibliográfica.

   1.1. Análisis de estándares de metamodelado vinculados a MDA, técnicas de metamodelado y herramientas de soporte para metamodelado.
   1.2. Análisis del framework ANTLR para generar analizadores léxico y sintáctico.

2. Especificación del lenguaje NEREUS en EBNF

3. Desarrollo de un parser para el lenguaje NEREUS.

4. Desarrollo de un traductor del lenguaje NEREUS al lenguaje CASL

5. Desarrollo de casos de estudio que validen especificaciones de metamodelos MOF simplificados.

6. Redacción del informe final

Grado de avance

Se cuenta con un significativo avance en las etapas 1, 2 y 3 previéndose la finalización de las etapas 3, 4, 5 y 6 en el año 2014.

5. BIBLIOGRAFÍA

[1] Aho, Alfred, Lam, Monica, Sethi, Ravi, Ullman, Jeffrey. Compilers: Principles, Techniques, and Tools, Pearson Education, 2006.

[2] Astesiano, E., Kreowski, H:, Krieg-Bruckner (eds) Algebraic Foundations of System Specification. IFIP State of Art Reports, Springer, 1999.

[3]Bidoit, Michel, Mosses, Peter. Casl User Manual - Introduction to Using the Common Algebraic Specification Language**.** Lecture Notes in Computer Science 2900, Springer-Verlag, 2004,

[4]Brambilla, Marco, Cabot, Jordi, Wimmer, Manuel. Model Driven Engineering in Practice. Morgan & Claypool Publishers, 2012.

[5]CASE MDA. Committed Companies and Their Products. www.omg.org/mda/committed-products.htm, 2014.

[6] COFI. The Common Framework Initiative for algebraic specification and development http://www.informatik.uni-bremen.de/cofi/wiki/index.php/CoFI 2014

[7] EMF- Eclipse Modeling Framework. www.eclipse.org,2014

[8] Favre, Liliana. A Formal Foundation for Metamodeling**.** Ada-Europe 2009: Lecture Notes in Computer Science 5570, Springer-Verlag, pág. 177-191, 2009.

[9] Jackson,Ethan, Levendovszky, Tihamer , Balasubramanian, Daniel. Reasoning about Metamodeling with Formal Specifications and Automatic Proofs, in MODELS 2011, 2011

[10] MDA The Model-Driven Architecture. from http://www.omg.org/mda/, 2014

[11] OMG –Object Mangement Group. www.omg.org, 2014

[12] OMG. Omg Meta Object Facility (MOF) core specification, version 2.4.1, agosto 2011.

[13] OMG. Omg Object Constraint Language (OCL), version 2.3.1, 2012.

[14] OMG QVT: MOF 2.0 Query, View, Transformation. Version 1.1, OMG Document Number : formal/2011-01-01. http://www.omg.org/spec/QVT/1.1/

[15] Parr, Terence. The Definitive ANTLR 4 Reference (1st ed.), Pragmatic Bookshelf, p. 328, ISBN 978-1-93435-699-9, 2013.

[16] UML 2011. Unified Modeling Language: Infrastructure. Version 2.4.1, OMG Specification formal/2011-08-05. http://www.omg.org/spec/UML/2.4.1/

Firma del alumno

Avalo la presente solicitud de evaluación

Firma del director