Utilización de Maude desde Eclipse Modeling Framework para la Gestión de Modelos[1]

Artur Boronat, José Iborra, José Á. Carsí, Isidro Ramos, Abel Gómez
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
Camí de Vera, s/n
46022 València
{aboronat | jiborra | pcarsi | iramos | agomez}@dsic.upv.es

Resumen

Los métodos formales proporcionan buenas propiedades para abordar problemas en Ingeniería del Software. Sin embargo, en muchos casos no se suelen aplicar en un ámbito industrial debido a prejuicios o malas experiencias. En este artículo, se presenta un caso de éxito de la aplicación de especificaciones algebraicas en un entorno industrial de modelado para dar soporte a la Gestión de Modelos. Esta disciplina es una nueva tendencia dentro de la Ingeniería de Modelos que trata a los modelos como ciudadanos de primer orden y que proporciona una serie de operadores genéricos para manipularlos. Se ha especificado algebraicamente un conjunto de operadores de este tipo utilizando el lenguaje Maude. Estos operadores se utilizan de forma visual desde Eclipse Modeling Framework (EMF). En este artículo se presenta el soporte que se ofrece para la interoperabilidad entre Maude y EMF en una herramienta de gestión de modelos.

1. Introducción

En la Ingeniería del Software aplicada a la industria, la aplicación de formalismos para dar soporte a herramientas industriales no ha sido siempre bien recibida. Pese a que existen casos de exitosa aplicación [1], una serie de mitos y creencias separa a la comunidad teórica de la industria. En [2] se plantea un interesante debate sobre los puntos críticos de este cisma, basándose en una década de discusiones sobre este tema. En este artículo se presenta un caso exitoso de aplicación de especificaciones algebraicas a una herramienta industrial de modelado, en el contexto de la Gestión de Modelos.

En la Ingeniería de Modelos, cualquier artefacto software puede ser tratado o representado como un modelo [3]: ontologías, modelos UML, esquemas relacionales, esquemas XML, etc. Un modelo describe una realidad física, abstracta o hipotética, recogiendo únicamente la información necesaria que permite conseguir unos objetivos específicos: generación de código, integración de aplicaciones, interoperabilidad entre aplicaciones, etc. Trabajar directamente sobre modelos aumenta el nivel de abstracción de estas tareas y permite automatizarlas. Las herramientas que dan soporte a este tipo de tareas normalmente lo consiguen de una forma específica para un determinado contexto de trabajo (ontologías, bases de datos relacionales, etc) o para una determinada tecnología (integración de modelos UML con Rational Rose Integrator), generación de código a partir de herramientas de modelado (como Rational Rose, Visio), etc.

Dentro de la Ingeniería de Modelos, una nueva disciplina, llamada Gestión de Modelos [4], trata con modelos mediante una serie de operadores genéricos que permiten realizar tareas como las descritas anteriormente. Estos operadores constituyen una solución abstracta y reutilizable para trabajar sobre los modelos como ciudadanos de primer orden, independientemente del contexto o tecnología utilizada para representarlos.

[1] Este artículo ha sido financiado por el Proyecto Nacional de Investigación, Desarrollo e Innovación DYNAMICA TIC 2003-07804-C05-01.

Dada nuestra experiencia en la aplicación del formalismo de especificaciones algebraicas a la recuperación de sistemas legados [5, 6], se está desarrollando una herramienta que da soporte algebraico a este tipo de operadores genéricos desde un entorno de modelado industrial. Nuestra aproximación demuestra la falacia de los mitos basados en la poca productividad de las herramientas formales y en el aumento del coste del proceso software debido a su uso. Para ello hemos elegido un eficiente sistema de reescritura de términos, Maude [7], que ya ha sido utilizado en muchos ámbitos formales [8].

Nuestra herramienta de gestión de modelos, llamada MOMENT (MOdel manageMENT), utiliza el entorno Maude desde un entorno de modelado industrial, como es Eclipse Modeling Framework (EMF) [9]. Esta integración de un método formal en una herramienta industrial de desarrollo de software, combina los esfuerzos que se están realizando sobre ambas herramientas en direcciones divergentes: en Maude sobre aspectos teóricos, y en EMF sobre su aplicación a la Ingeniería del Software. De esta manera, evitamos el aislamiento de Maude en el ámbito industrial, debido a la premisa "construyelo y ya vendrán"[2], frecuentemente mantenida por los desarrolladores de métodos formales. Este hecho permite utilizar Maude para solucionar problemas reales en la Ingeniería del Software, sin limitarse únicamente a la solución de los llamados ejemplos de juguete.

Este artículo presenta el mecanismo que se ha desarrollado en el marco de MOMENT para conseguir una buena interoperabilidad entre Maude y EMF. La estructura del artículo es la siguiente: la Sección 2 presenta una visión global de la aplicabilidad de la Gestión de Modelos presentando un ejemplo que se utilizará a lo largo del trabajo; en la Sección 3, se realiza una discusión sobre herramientas de modelado en las tendencias más relevantes de la Ingeniería de Modelos; en la Sección 4, se presenta el marco conceptual de nuestra solución para la interoperabilidad entre Maude y EMF; en la Sección 5, se presenta cómo se da soporte para la proyección de modelos EMF sobre Maude; finalmente, en la sección 6 se presentan una serie de conclusiones y trabajos futuros.

[2] Del inglés, "built-it-and-they-will-come".

2. Visión global del framework MOMENT

En MOMENT los operadores de gestión de modelos [10] han sido especificados algebraicamente utilizando el formalismo Maude. Los modelos se especifican como conjuntos de elementos de forma independiente del metamodelo, de manera que los operadores pueden acceder a los elementos sin conocer la representación de un modelo. La interfaz de MOMENT está integrada en EMF, de manera que el formalismo de especificaciones algebraicas queda totalmente transparente al usuario.

Para ilustrar el funcionamiento de MOMENT, se indica un pequeño ejemplo de integración de esquemas XML. Para ello se ha definido una parte del metamodelo del lenguaje de definición XML (XSD), mostrado en la Figura 1 utilizando notación UML.

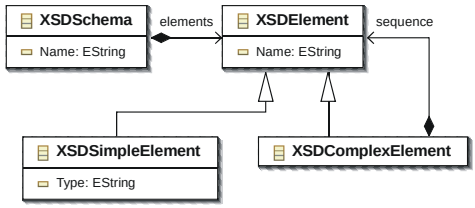

Figura 1. Parte del metamodelo XSD.

Utilizando el editor en forma de árbol que proporciona EMF, definimos los esquemas XML *A* y *B* en la Figura 2. Se aplica el operador *Merge* a ambos, obteniendo el esquema XML integrado *C* y dos modelos de trazas (map_{AC} y map_{BC}) que enlazan los elementos de los modelos de entrada con los elementos del modelo de salida. La invocación del operador es la siguiente: *<C, mapAC, mapBC> = Merge(A, B).*

Figura 2. Aplicación del operador Merge.

Para poder realizar la integración de los esquemas XML, estos deben ser traducidos a

términos en Maude, para que el operador Merge pueda aplicarse sobre ellos. En los siguientes apartados se indica como se consigue esta interoperabilidad entre EMF y Maude mediante MOMENT.

3. Herramientas de modelado

En esta sección se indican algunas herramientas que dan soporte para la Ingeniería de Modelos, teniendo en cuenta las últimas tendencias. De este modo, se justifica el entorno de modelado elegido para nuestra aproximación de gestión de modelos.

Debido a la gran variedad de artefactos software, la interoperabilidad con herramientas de modelado de cualquier índole constituye una condición necesaria para que una herramienta de gestión de modelos tenga éxito. La iniciativa Model-Driven Architecture (MDA) [11] proporciona una serie de estándares que permiten compartir metainformación entre herramientas de modelado (XMI) y repositorios de metainformación (MOF).

Algunas herramientas que dan soporte a este enfoque son Eclipse Modeling Framework y NetBeans MetaData Repository. Eclipse Modeling Framework (EMF) es un entorno de modelado que proporciona facilidades para la generación de código. EMF puede ser utilizado como un marco de metamodelado donde el lenguaje común es un subconjunto de UML. NetBeans MetaData Repository da soporte tecnológico al estándar MOF constituyendo un repositorio de metainformación que puede ser modelada en UML a través de entornos visuales de modelado.

Una aproximación similar es seguida en la computación integrada en modelos (MIC – Model Integrated Computing) [12], con herramientas como Generic Modeling Environment y MetaEdit+. Todas estas herramientas constituyen un marco de metamodelado basadas en un lenguaje abstracto para definir la sintaxis, semántica y visualización de lenguajes específicos a dominio.

Siguiendo el enfoque de la Ingeniería Específica a Dominio [13], un nuevo paradigma de desarrollo de software está emergiendo con las llamadas Factorías de Software (Software Factories [14]). Esta aproximación se basa en la definición de lenguajes específicos a dominio para abordar el desarrollo de software en un contexto determinado, aumentando la calidad, la productividad y la automatización del proceso de desarrollo. La apuesta de Microsoft en este campo se enmarca en la herramienta Domain Specific Language tools, aún en fase preliminar. También existen propuestas de nuevas herramientas como Meta Programming System, y Graphical Modelling Framework (que funcionará sobre EMF).

Entre todas estas herramientas, hemos elegido EMF como entorno de modelado para nuestra herramienta de Gestión de Modelos por su situación dentro del marco de la Ingeniería de Modelos. EMF permite tratar con gran variedad de artefactos software, como esquemas XML, modelos UML (definidos en entornos visuales de modelado como Rational Rose), esquemas relacionales (a través de Rational Rose), y ontologías, entre otros. Además, EMF es utilizada por las principales herramientas de IBM, aportando una visión industrial a nuestro enfoque de Gestión de Modelos.

4. Espacios tecnológicos: EMF y Maude

El concepto de espacios tecnológicos fue introducido por Kurtev et al. en la discusión sobre el enlace de tecnologías heterogéneas [15]. Un espacio tecnológico (ET) es un contexto de trabajo en el que se dispone de un conjunto de conceptos bien definidos, una base de conocimiento, herramientas, y una serie de posibilidades de aplicación específicas [16]. Por ejemplo, en este artículo tratamos con los ETs de Eclipse Modeling Framework y de Maude.

En esta sección, se indica el marco conceptual que se ha utilizado para representar artefactos software mediante EMF y cómo estos son proyectados sobre Maude, con el objetivo de utilizar los operadores algebraicos de MOMENT sobre ellos.

4.1. Marco conceptual para la representación de artefactos software en Maude

Siguiendo un enfoque de Ingeniería de Modelos, para tratar con artefactos software utilizamos la terminología que define el estándar Meta-Object Facility de la iniciativa MDA. Este estándar presenta una arquitectura de cuatro capas de modelado que permite clasificar artefactos

software con diferente propósito: M3 (metametamodelos), M2 (metamodelo), M1 (modelo), M0 (sistema real).

Una estrategia para trabajar con metamodelos consiste en definir una sintaxis básica en el nivel M3, que pueda ser utilizada para definir artefactos software en niveles inferiores. En EMF, el metametamodelo se llama Ecore y proporciona una serie de primitivas de modelado: un subconjunto del diagrama de clases del metamodelo UML. Estas primitivas se utilizan para definir metamodelos en el nivel M2, constituyendo un paradigma de modelado. Como por ejemplo, el lenguaje de definición de esquemas XML (XML Schema Definition language - XSD).

Los elementos de un metamodelo son utilizados como tipos para definir los elementos que constituyen un modelo en el nivel M1. En el caso del metamodelo XSD, un modelo es un esquema XML específico. Los elementos de un modelo también se comportan como tipo para definir información en el nivel M0 de la arquitectura MOF. Por ejemplo, un esquema XML define los elementos que se pueden utilizar en un documento XML.

4.2. Proyecciones de artefactos software EMF sobre Maude

Un ET se caracteriza por el soporte tecnológico que se proporciona a un determinado paradigma de modelado. Cada paradigma de modelado se organiza entorno a un metametamodelo común y persigue unos objetivos específicos.

El ET EMF se caracteriza por las facilidades que ofrece para representar una buena variedad de artefactos sofware como modelos y por su interoperabilidad con otras herramientas industriales de modelado. El ET Maude se caracteriza por las ventajas que aporta el formalismo de especificaciones algebraicas: abstracción, subtipado, modularización, genericidad mediante parametrización, etc. Este ET también puede ser visto como un paradigma de modelado[3], considerando el lenguaje Maude como el lenguaje de definición de metamodelos en el nivel M3. En el nivel M2, los metamodelos son los módulos que proporcionan especificaciones algebraicas Maude.

Una especificación algebraica representa un metamodelo, proporcionando la descripción sintáctica de las primitivas (llamadas constructores en el campo de las especificaciones algebraicas), necesarias para especificar un artefacto software en el nivel M1. Cuando la especificación algebraica es interpretada como álgebra, se obtiene la visión como tipo del metamodelo, donde los constructores se pueden utilizar para definir artefactos software en el nivel M1. Éstos son representados sintácticamente como términos, representando la información en forma de árbol.

Para manipular modelos EMF con los operadores algebraicos de MOMENT se han definido una serie de proyecciones entre ambos ETs (ver Figura 3). Estas proyecciones permiten representar un modelo como un término algebraico, manipularlo desde Maude, y devolverlo como un modelo EMF.

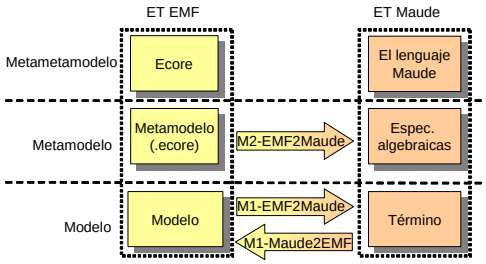

Figura 3. Enlaces entre el ET EMF y el ET Maude.

4.3. Interoperabilidad en el nivel M2

En el nivel de metamodelos se establece un enlace unidireccional que proyecta un metamodelo EMF sobre el ET Maude, obteniendo una especificación algebraica. De este modo, un metamodelo se interpreta como un álgebra que proporciona los constructores necesarios para definir modelos y

[3] Para simplificar el discurso, utilizamos una visión sintáctica de las especificaciones algebraicas y no tenemos en cuenta las capacidades de reflexión que Maude proporciona. Otra visión es la semántica, donde las capacidades de reflexión de Maude sitúan el Álgebra Universal como metametamodelo en el nivel M3.

las operaciones necesarias para manipularlos, en el contexto de la Gestión de Modelos.

Este enlace es unidireccional pues los metamodelos se especifican mediante herramientas visuales de modelado a través de EMF (el nombre que hemos asignado a este enlace es M2-EMF2Maude). Este hecho permite hacer transparente el uso del formalismo Maude al usuario final del framework MOMENT.

MOMENT trabaja directamente sobre un álgebra de operadores genéricos de manipulación de modelos, especificados en el módulo MOMENT-OP(X::TRIV). Estos operadores pueden ser adaptados a un metamodelo específico haciendo uso de las capacidades de parametrización que ofrece Maude, basándose en el concepto formal de Pushout [17] de teoría de categorías.

En el diagrama del mecanismo de paso de parámetros[4] (Figura 4), TRIV constituye el parámetro formal del módulo parametrizado MOMENT-OP(X::TRIV). La especificación sigXSD constituye el parámetro real para el módulo parametrizado. sigXSD proporciona los constructores correspondientes a las primitivas de un metamodelo específico. Esta especificación algebraica está relacionada con el parámetro formal mediante la vista vXSD. Como ejemplo se ha utilizado el metamodelo XSD simplificado. En él, a partir de la metainformación[5] que describe la clase XSDSimpleElement, que indica cómo definir un elemento simple en un esquema XML, se obtiene un constructor que permite definir un elemento simple en un esquema XML como un término del álgebra.

El módulo MOMENT-OP(vXSD) representa la instaciación del módulo parametrizado con el parámetro real sigXSD. Esta especificación valor es importada por el módulo spXSD, en el que se extiende la presentación axiomática de los operadores genéricos adaptándola a un metamodelo específico. Por ejemplo, el operador *Merge*, cuando se utiliza en el metamodelo XSD, permite integrar esquemas XML. Para definir este tipo de integración de forma más precisa, se pueden añadir relaciones de equivalencia que tengan en cuenta las primitivas del metamodelo XSD: elemento simple, elemento complejo, etc. Estas relaciones de equivalencia específicas al metamodelo XSD se añaden al módulo spXSD en forma de axiomas. La especificación algebraica resultante constituye el álgebra de un metamodelo. El módulo spXSD también es generado automáticamente por la herramienta MOMENT a partir de la especificación de los nuevos axiomas en interfaces visuales[6].

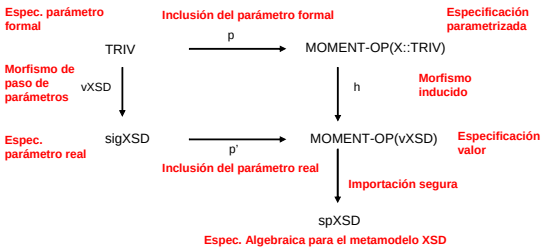

Figura 4. Diagrama del mecanismo de paso de parámetros en Maude.

4.4. Interoperabilidad en el nivel M1

Existe otro tipo de enlace entre el ET EMF y el ET Maude en el nivel de modelos. Este enlace es bidireccional y consta de dos tipos de proyecciones:

- M1-EMF2Maude: Este mecanismo proyecta un modelo EMF, definido mediante un metamodelo EMF, sobre el ET Maude como un término. Para proyectar un modelo sobre el ET Maude, la herramienta MOMENT consulta el correspondiente metamodelo y obtiene el constructor de la correspondiente especificación algebraica, que es necesario

[4] En la figura, p y p' son morfismos que indican inclusión de especificaciones algebraicas, y h es el morfismo inducido a partir de la vista vXSD, que obtiene los elementos de MOMENT-OP(vXSD) a partir de los elementos del módulo parametrizado.

[5] Expresada como una instancia de la clase EClass del metametamodelo Ecore.

[6] El objetivo del artículo se centra en el soporte que MOMENT ofrece para la proyección de artefactos software de EMF a Maude. Como la generación del módulo spXSD concierne más a la semántica de las operaciones de manipulación de modelos, nos centramos en la obtención del parámetro real para el álgebra genérica de operadores (en el caso del metamodelo XSD, es el módulo sigXSD).

para especificar el término de forma automática.

- M1-Maude2EMF: Este mecanismo proporciona la proyección inversa a la anterior, obteniendo un modelo EMF a partir de un término Maude. En este paso, cuando la herramienta MOMENT lee un término que representa un modelo, determina las primitivas del metamodelo EMF que debe utilizar para construir el modelo EMF a partir de los símbolos de los constructores utilizados en el término. Con estas primitivas se construye el modelo dinámicamente y se persiste en formato XMI.

Los enlaces, que han sido descritos entre el ET EMF y el ET Maude, permiten la aplicación de operadores algebraicos sobre modelos definidos de forma gráfica mediante entornos industriales de modelado. Por ejemplo, supongamos que se desea realizar la integración de dos esquemas XML, cuyo metamodelo ha sido definido mediante Ecore. El proceso seguido es el siguiente:

1. Se obtiene la especificación algebraica spXSD a partir del metamodelo XSD.
2. Se proyectan los esquemas XML A y B a términos del álgebra, interpretada a partir de spXSD.
3. Se aplica el operador Merge a ambos términos y se obtiene el término resultante mediante el mecanismo de reducción de Maude. Como resultado de la operación de integración se obtiene el término C[7], que representa el esquema XML integrado.
4. Finalmente, el término C es proyectado al ET EMF como un modelo.

5. Soporte tecnológico para la interoperabilidad entre EMF y Maude

En esta sección, se detalla la implementación del soporte para la interoperabilidad entre el ET Maude y el ET EMF, que ofrece MOMENT. En primer lugar, se indica cómo se ha integrado Maude en la plataforma Eclipse. En segundo lugar, se detalla la implementación de los enlaces entre ETs presentados conceptualmente en la sección anterior.

5.1. Maude en Eclipse

Para hacer uso de las características de Maude, se ha desarrollado dos plug-ins que nos permiten su uso directamente desde el entorno de Eclipse.

El primero consta de todas las funcionalidades para configurar, ejecutar, e interactuar con Maude desde Java. Este plug-in incluye una ventana de configuración integrada en las preferencias de Eclipse para indicar dónde se encuentran los ficheros necesarios, así como otras opciones. El plug-in, a su vez, proporciona dos APIs para invocar a Maude desde el código de una aplicación: de forma interactiva o por lotes.

El segundo plug-in se trata de un editor con coloreado de sintaxis que facilita la escritura de programas y su ejecución directamente en una consola de Maude. Proporciona los correspondientes asistentes para la creación de ficheros nuevos, así como una barra de herramientas para interactuar con el proceso.

5.2. Soporte para la interoperabilidad en el nivel de metamodelos

El enlace del nivel M2 obtiene la especificación algebraica correspondiente a un metamodelo especificado en Ecore. En concreto, para cada tipo del metamodelo se incluye un sort, y para cada tipo no abstracto[8] se incluye en la especificación algebraica un constructor.

La obtención de la especificación algebraica de un metamodelo se ha automatizado mediante técnicas generativas a través de patrones. Con este objetivo, se ha utilizado el motor de plantillas Velocity [18]. Un motor de plantillas toma como entrada una plantilla y un contexto para la plantilla, que proporciona información para generar código, como la resolución de referencias.

En el contexto de la proyección de metamodelos a Maude se ha definido una plantilla que hace posible la generación automática de la

[7] El operador Merge también produce dos modelos de trazabilidad que relacionan los modelos de entrada A y B con el modelo de salida C, respectivamente. Ambos modelos se han obviado para simplificar el ejemplo.

[8] En el contexto de MOF y EMF, el concepto de tipo abstracto incluye las interfaces y las clases abstractas.

especificación algebraica correspondiente. Para realizar la proyección de un metamodelo en el espacio tecnológico de Maude se siguen los siguientes pasos:

1. Previamente a la generación, el metamodelo es analizado y procesado para producir una estructura más cercana a la representación en la plantilla destino. El procesado incluye: la clasificación de los tipos en abstractos y concretos, la definición de la jerarquía de sorts, y la obtención de la signatura de los constructores.
2. El motor de plantillas recibe el resultado de este procesado y la plantilla a utilizar, y produce la correspondiente especificación algebraica del metamodelo.

En la Figura 5, se muestra una captura de pantalla en la que aparece la especificación generada a partir del metamodelo XSD del ejemplo, en la interfaz del editor Maude.

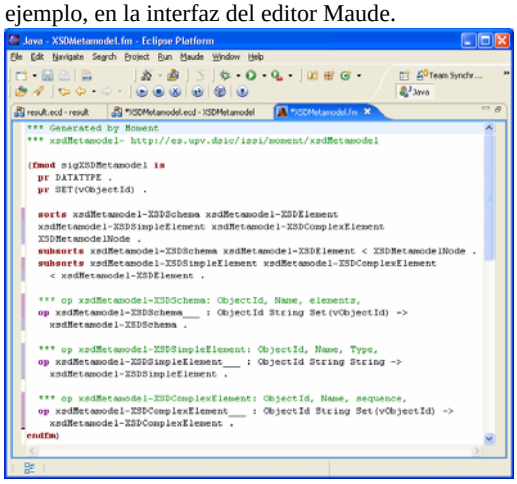

Figura 5. Especificación sigXSD generada a partir del metamodelo XSD simplificado.

5.3. Soporte para la interoperabilidad en el nivel M1: M1-EMF2Maude

La proyección de un modelo en el espacio tecnológico de Maude equivale a un conjunto de términos de la especificación algebraica correspondiente al metamodelo utilizado. Para este mecanismo se ha utilizado la misma técnica generativa que en el caso anterior. Por lo tanto el proceso seguido es similar, y en este caso las responsabilidades del preprocesado son las siguientes:

- Sustitución de tipos de datos de EMF por tipos de datos de Maude.
- Identificación de todos los metamodelos que intervienen en la definición del modelo
- Generación de identificadores únicos. Cada término generado a partir de un elemento de un modelo EMF tiene asignado un identificador, que es utilizado para dar soporte a las referencias en Maude.
- Obtención del constructor de la especificación algebraica, que corresponde al tipo de un elemento del modelo.

5.4. Soporte para la interoperabilidad en el nivel M1: M1-Maude2EMF

El problema de la obtención de modelos EMF a partir de términos algebraicos se ha abordado como un problema de análisis de lenguaje, teniendo en cuenta la sintaxis Maude para especificar términos.

Para generar los analizadores léxico, sintáctico y semántico requeridos se ha empleado el generador de parsers ANTLR [19], cuyo funcionamiento es similar a los conocidos Bison y Flex. La mayor parte de la lógica presente en el proceso de análisis se concentra en el analizador semántico, encargado de recorrer el Arbol Abstracto (Abstract Syntax Tree, AST), que está formado por representantes de los términos algebraicos procedentes del código fuente, y de ir construyendo el modelo en EMF a medida que se analice. Para cada término el proceso es el siguiente:

1. A partir del símbolo del constructor del término se identifica el metamodelo que contiene al tipo que representa a este término, y se carga la clase del elemento representado por el término. El analizador semántico crea una instancia dinámicamente, que ya es la proyección en EMF del término algebraico.
2. Seguidamente se rellenan los atributos de la instancia creada con los valores de los subtérminos encontrados en el término. Las referencias se dejan pendientes.
3. Una vez se han procesado todos los términos se resuelven las referencias entre ellos, de forma similar al proceso seguido con los atributos.

6. Conclusiones

En este artículo se ha presentado una visión global de la herramienta MOMENT, y el marco conceptual en el que se está trabajando dentro de la Ingeniería de Modelos. Se ha presentado el soporte para la interoperabilidad entre los espacios tecnológicos EMF y Maude, que permite aplicar operadores definidos algebraicamente a modelos definidos mediante herramientas visuales de modelado. Además, este hecho permite combinar los esfuerzos que se están realizando actualmente sobre cada una de estas herramientas, en el campo industrial de EMF y en el campo teórico de Maude.

Como trabajo futuro, se está desarrollando el soporte para definir las relaciones existentes entre modelos de forma visual. El objetivo consiste en proporcionar una herramienta que permita definir operadores de gestión de modelos algebraicamente y que permita su aplicación desde una herramienta visual de modelado.

Agradecimientos

A Francisco Durán, por facilitarnos la documentación y la implementación de Maude Workstation, en el que nos hemos basado para ejecutar Maude sobre Windows.

Referencias

[1] Bowen, J.P., Hinchey, M.G. Seven More Myths of Formal Methods. IEEE Software. July 1995 (Vol. 12, No. 4).

[2] Gogolla, M. Benefits and Problems of Formal Methods. *Proc. 9th Int. Conf. Reliable Software Technologies Ada-Europe (RST'2004)*, pages 1-15. Springer, Berlin, LNCS 3063, 2004.

[3] Bézivin, J.: On the Unification Power of Models, in: "Software and System Modeling (SoSym)", 2004, vol. 3, no 4.

[4] Bernstein, P.A., Levy, A.Y., Pottinger, R.A.: A Vision for Management of Complex Models. MRS Tech. Rep. MSR-TR-2000-53, (short version in SIGMOD Record 29, 4 (Dec. '00)).

[5] Boronat, A., Pérez, J., Carsí, J. Á., Ramos, I.: Two experiencies in software dynamics. Journal of Universal Science Computer. Special issue on Breakthroughs and Challenges in Software Engineering. Vol. 10 (issue 4). April 2004.

[6] Boronat, A., Carsí, J.Á., Ramos, I.: Automatic Reengineering in MDA Using Rewriting Logic as Transformation Engine. IEEE Computer Society Press. CSMR 2005. Manchester, UK. 2005.

[7] Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Quesada, J.F.: Maude: specification and programming in rewriting logic. Theoretical Computer Science, 285(2):187-243, 2002.

[8] Martí-Oliet N., Meseguer, J. Rewriting logic: Roadmap and bibliography. Theoretical Computer Science, 285(2):121–154, 2002.

[9] El sitio web de EMF: http://download.eclipse.org/tools/emf/scripts/

[10] Bernstein, P.A: Applying Model Management to Classical Meta Data Problems. pp. 209-220, CIDR 2003.

[11] OMG Model-Driven Architecture. http://www.omg.org/mda/

[12] Sztipanovits J. y Karsai G., Model-Integrated Computing, Computer, Apr. 1997, pp. 90-92.

[13] Czarnecki, K., Eisenecker, U.: Generative Programming: Methods, Tools, and Applications.Addisson-Wesley (2000). ISBN 0-201-30977-7, pag. 267-304.

[14] Greenfield, J., Short, K., Cook, S., Kent, S., Crupi, J. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Wiley. 2004. ISBN: 0-471-20284-3

[15] Kurtev, I., Bézivin, J., Aksit, M.: Technological Spaces: An Initial Appraisal. Int. Federated Conf. (DOA, ODBASE, CoopIS), Industrial track, Irvine, 2002.

[16] Bézivin, J., Devedzic, V., Djuric, D., Favreau, J.M., Gasevic, D., Jouault, F.: An M3-Neutral infrastructure for bridging model engineering and ontology engineering. INTEROP-ESA. Switzerland. 2005.

[17] Ehrig, H., Mahr, B.: Fundamentals of Algebraic Specification 1. Springer-Verlag Berlin Heidelberg New York Tokio (1985). ISBN: 3-540-13718-1.

[18] El motor de plantillas Apache Velocity: http://jakarta.apache.org/velocity

[19] El generador de parsers ANTLR: http://www.antlr.org