Transformación de modelos con Eclectic

Jesús Sánchez Cuadrado[1]

Universidad Autónoma de Madrid (Spain)
jesus.sanchez.cuadrado@um.es

Abstract. Las transformaciones de modelos son un elemento clave del Desarrollo de Software Dirigido por Modelos. En los últimos años se han propuesto varios lenguajes de transformación de diferente naturaleza, siendo cada uno de ellos adecuado para un determinado tipo de tarea de transformación. Sin embargo, una transformación compleja normalmente implica abordar una serie de sub-problemas que corresponden a diferentes estilos de transformación, y por tanto no toda la transformación puede desarrollarse de forma natural en el lenguaje elegido.
En esta demostración se presentará el entorno de transformación de modelos *Eclectic*, que trata de abordar el desarrollo de transformaciones de modelos ofreciendo una familia de lenguajes de transformación. Cada lenguaje tiene como objetivo abordar un determinado tipo de transformaciones, y está específicamente diseñado para ello. La demostración se ilustrará con un ejemplo de aplicación que utiliza diferentes lenguajes, se mostrará el entorno de desarrollo y se comentarán características de la aproximación tales como interoperabilidad entre lenguajes e integración con programas Java.

1 Motivación

Las transformaciones de modelos son un elemento clave del Desarrollo de Software Dirigido por Modelos (DSDM), puesto que permiten automatizar la manipulación de los modelos. En los últimos años se han propuesto varios lenguajes de transformación de diferente naturaleza, tales como ATL [5], RubyTL [3], QVT [8] o Kermeta [6]. Las taxonomías de lenguajes de transformación propuestas por Czarnecki [4] y Mens [7] evidencian que cada lenguaje proporciona una serie de características que lo hacen más adecuado para abordar cierto tipo de tarea de transformación. Hasta el momento el diseño de lenguajes de transformación ha seguido básicamente dos aproximaciones: a) mantener el lenguaje simple, declarativo y orientado a cierto tipo de transformaciones, b) complicar el lenguaje añadiendo características para que tenga un ámbito de aplicación más amplio. El problema de la primera aproximación es que la aplicabilidad del lenguaje es limitada, y normalmente solo permite que el lenguaje se use para abordar transformaciones simples, mientras que la segunda aproximación complica el diseño original con lo que las transformaciones tienen tendencia a ser ilegibles a medida que son más complicadas, debido principalmente a que las características declarativas del lenguaje no son suficientes.

Para abordar esta problemática se propone un diseño alternativo, basado en una familia de lenguajes de transformación [2].

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 557-560, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

2 Eclectic: una familia de lenguajes de transformación

En sección se describe brevemente las características de la familia de lenguajes de transformación que se ha definido, llamada Eclectic, así como su estado actual y el trabajo futuro.

La idea principal de esta aproximación es que una transformación compleja podría dividirse en varias tareas de transformación de tamaño más pequeño, que podrían resolverse utilizando lenguajes específicamente diseñado para ellas. Se pretende así mejorar la declaratividad y la "intencionalidad" de las definiciones de transformación, puesto que los lenguajes que se usarían no son de propósito general, sino que tienen las características (y solo esas) que hacen falta resolver ese tipo de tarea.

Para alcanzar ese objetivo es imprescindible que los diferentes lenguajes que componen Electic sean interoperables, así como disponer de mecanismos de composición que permitan especificar cómo los resultados obtenidos por cada una de las transformaciones contribuyen al resultado final. En estos momentos se han investigado los siguientes elementos de la aproximación:

- *Interoperabilidad* entre lenguajes de transformación heterogéneos. Este objetivo se ha conseguido definiendo un lenguaje intermedio, IDC (Intermediate Dependency Code), al cual compilan el resto de lenguajes. Algo más de información acerca de IDC puede encontrarse en [9].
- *Mecanismos de composición* en transformación de modelos. El lenguaje IDC soporta los siguientes: alimentar reglas o patrones con valores obtenidos mediante cierto recorrido de un modelo, resolución de referencias de elemento origen a destino, decorar metaclases con métodos virtuales y configuración de la ejecución de transformaciones.
- *Tipos de transformaciones*. ¿Qué tipos de transformaciones debería soportar la familia de lenguajes que se está construyendo? Hasta ahora se han identificado cinco dominios y se ha implementado un lenguaje para cada uno de ellos:
  - Definición de correspondencias o *mappings*. Tiene como objetivo resolver heterogeneidades entre partes de meta-modelos semánticamente equivalentes.
  - Transformaciones dirigidas por la estructura del modelo destino. Suelen darse cuando se compila un modelo de más alto nivel de abstracción a otro de más bajo nivel, y requiere inicializar varios objetos a partir de cierto objeto origen.
  - Cálculo de atributos, al estilo de las definiciones dirigidas por la sintaxis [1], donde el valor de un atributo para un elemento depende de los valores de los atributos de otros elementos accesibles a partir del primero.
  - Búsqueda de patrones complejos, cuyo resultado debe utilizarse para alimentar las reglas de algunos de los lenguajes de la familia.
  - Composición de transformaciones, para crear una transformación que aborde el problema global.

– *Integración* con lenguajes de programación de propósito general. Para abordar este objetivo se ha creado un compilador que genera *bytecode* para la máquina virtual de Java (JVM), de manera que las transformaciones escritas con Eclectic puede integrarse de manera natural con programas Java. En este aspecto es importante considerar también la integración a nivel de entorno de desarrollo, que se ha realizado para Eclipse.

La versión actual de Eclectic es todavía una prueba de concepto pero ya se ha conseguido crear una implementación que demuestra que la aproximación es viable. Está disponible en `http://sanchezcuadrado.es/projects/eclectic` para su descarga. La Figura 1 es una captura de pantalla del entorno que se está construyendo para Eclipse, en el que se están editando dos transformaciones, que tratan con modelos UML, OCL y Java: una con el lenguaje de cálculo de atributos y otra con el lenguaje de *mappings*.

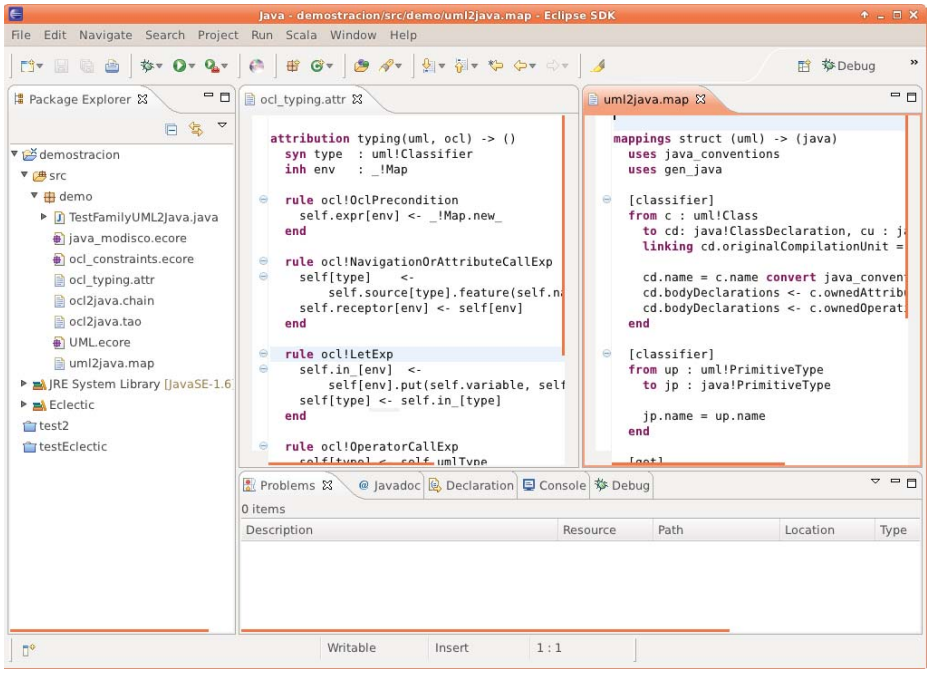

Fig. 1: Entorno de desarrollo en Eclipse

Como trabajo futuro se plantean las siguientes cuestiones:

– Seguir investigando y refinando los lenguajes que componen Eclectic y sus características, así como escribir guías que orienten al desarrollador a la hora de elegir el lenguaje más apropiado y aplicarlo.
– Implementar casos de estudio de relativa complejidad que permitan determinar si esta aproximación presenta una mejora real con respecto a utilizar un único lenguaje de transformación.

– Estudiar la integración de lenguajes de transformación específicos del dominio, posiblemente en forma de librerías.
– Aprovechar las posibilidades de la JVM para mejorar el rendimiento.

3 Demostración

Con la presente demostración se pretende dar a conocer Eclectic y recabar la opinión de la comunidad sobre esta aproximación así como posibles mejoras o ideas. Para ello se prepará un póster que resuma las características de Eclectic y motive su utilización, así como una serie de ejemplos de aplicación para ser mostrados a los interesados en tener más detalles.

Agradecimientos. Este trabajo ha sido financiado por el Ministerio de Educación y Ciencia (TIN2011-24139) y la Comunidad de Madrid (S2009/TIC-1650).

References

1. A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: principles, techniques, and tools*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.
2. J. S. Cuadrado. Towards a family of model transformations languages. In *Proc. of the 5th International Conference on Model Transformati on (ICMT 2012, to appear)*, pages 260–275, LNCS 7307, 2012. Springer.
3. J. S. Cuadrado, J. G. Molina, and M. Menarguez. RubyTL: A Practical, Extensible Transformation Language. In *2nd European Conference on Model Driven Architecture*, volume 4066, pages 158–172. Lecture Notes in Computer Science, June 2006.
4. K. Czarnecki and S. Helsen. Feature-based survey of model transformation approaches. *IBM Syst. J.*, 45:621–645, July 2006.
5. F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev. ATL: A model transformation tool. *Science of Computer Programming*, 72(1-2):31 – 39, 2008. See also `http://www.emn.fr/z-info/atlanmod/index.php/Main_Page`. Last accessed: Nov. 2010.
6. Kermeta. `http://www.kermeta.org/`.
7. T. Mens and P. Van Gorp. A taxonomy of model transformation. *Electron. Notes Theor. Comput. Sci.*, 152:125–142, March 2006.
8. OMG. Final adopted specification for MOF 2.0 Query/View/Transformation, 2005. www.omg.org/docs/ptc/05-11-01.pdf.
9. J. Sanchez Cuadrado. Compiling ATL with Continuations. In *Proc. of 3rd International Workshop on Model Transformation with ATL (MtATL-2011)*, pages 10–19. CEUR-WS, 2011.