Creación automática de editores gráficos con Eugenia

Francisca Rosique Contreras, Pedro Sánchez Palma
División de Sistemas e Ingeniería Electrónica (DSIE)
Universidad Politécnica de Cartagena,
Campus Muralla del Mar s/n, Cartagena E-30202, Spain
paqui.rosique@upct.es, pedro.sanchez@upct.es

Resumen. *En los procesos de desarrollo de software dirigido por modelos, en general, es necesario definir modelos con un alto nivel de detalle, que suelen resultar modelos complejos y difíciles de manejar. El uso de editores gráficos ayudan a omitir detalles innecesarios y proporcionan una visión global del modelo. Para este fin suelen utilizarse editores basados en arcos y nodos (o diagramadores) que proporcionan una visión general del modelo. Siendo otro artefacto software más, no sólo es posible si no recomendable, aplicar los principios de la Ingeniería Dirigida por Modelos al desarrollo de estos editores con el objetivo de automatizar al máximo posible al proceso de desarrollo.* EuGENia *permite automatizar el proceso de generación de estos editores.*

1. Introducción

En los procesos de desarrollo de software dirigido por modelos, en general, es necesario definir modelos con un alto nivel de detalle, que suelen resultar modelos complejos y difíciles de manejar. El uso de editores gráficos ayudan a omitir detalles innecesarios y proporcionan una visión global del modelo. Para este fin suelen utilizarse editores basados en arcos y nodos (o diagramadores) que proporcionan una visión general del modelo. Siendo otro artefacto software más, no sólo es posible si no recomendable, aplicar los principios de la Ingeniería Dirigida por Modelos al desarrollo de estos editores con el objetivo de automatizar al máximo posible al proceso de desarrollo.

2. EUGENIA

Eugenia se integra dentro del proyecto Epsilon de Eclipse. Epsilon es una plataforma para la construcción de Lenguajes Específicos del Dominio (DSLs) [1] para tareas relacionadas con la gestión de modelos que incluye varios lenguajes para tareas como la transformación (ETL) [2] o la validación de modelos (EVL) [3]. A su vez, la base de la plataforma es EOL (Epsilon Object Language) [4], un lenguaje imperativo que permite crear, consultar y modificar modelos EMF [5].

Además de esta familia de lenguajes, Epsilon proporciona varias facilidades para llevar a cabo tareas comunes en el marco de la Ingeniería de Desarrollo Dirigida por Modelos. Una de ellas es EuGENia [6], una herramienta para simplificar el desarrollo de editores GMF.

A partir del metamodelo ecore, o su especificación textual con Emfatic [7], EuGENia genera automáticamente los modelos gmfgraph, gmftool y gmfmap. Para generar estos modelos, es necesario haber anotado previamente el metamodelo utilizando un conjunto de anotaciones predefinidas. EuGENia utiliza dichas anotaciones para saber cuál queremos que sea la representación gráfica de un elemento concreto o el control correspondiente en la paleta de herramientas. No obstante, un conjunto predefinido de anotaciones no suele ser suficiente para que el editor generado tenga exactamente las características y comportamiento deseados.

Por ello, EuGENia es capaz también de procesar tres ficheros EOL (Ecore2GMF.eol, FixGenModel.eol y FixGMFGen.eol) que permiten personalizar el editor.

La Figura 1 resume el proceso de desarrollo para generar un diagramador con EuGENia.

Nótese que los pasos 1, 2 y 3 son los mismos que se deben dar en la creación de un editor usando GMF. A partir del siguiente paso 4 es donde se diferencian.

4. Definir los ficheros de personalización codificados en lenguaje EOL.
5. Generar, a partir del modelo ecore y los ficheros de personalización, los modelos GMF (gmfgraph, gmftool y gmfmap).
6. Sincronizar los modelos ecore y genmodel.
7. Generar, a partir del modelo de mapeo, el modelo generador de GMF (gmfgen).
8. Sincronización de los modelos ecore y gmfgen.
9. Generar el código Java que implementa el diagramador.

Así, EuGENia no sólo reduce el número de modelos que debe definir el desarrollador, sino que, además permite la reutilización de los ficheros de personalización (o al menos parte de ellos) en diferentes desarrollos. Sin embargo, a pesar de estas mejoras el proceso de generación no es completamente automático, pues el desarrollador

sigue siendo el encargado de generar el editor tree-based de EMF, el modelo generador de GMF (GMFgen) y el código Java que implementa el diagramador. Además, el uso de Eugenia añade dos nuevas tareas al proceso: la sincronización de los modelos genmodel y gmfgen con el metamodelo ecore. No obstante, las mejoras aportadas por EuGENia sirven de base para la propuesta de este trabajo.

Estos pasos son realizados de forma transparente para el usuario, que tan sólo debe realizar una única acción, hacer click sobre la opción EuGENia que aparece en el menú contextual del fichero .emf.

En el siguiente apartado se presentan las principales anotaciones que pueden realizarse sobre el fichero .emf que son soportadas por Eugenia.

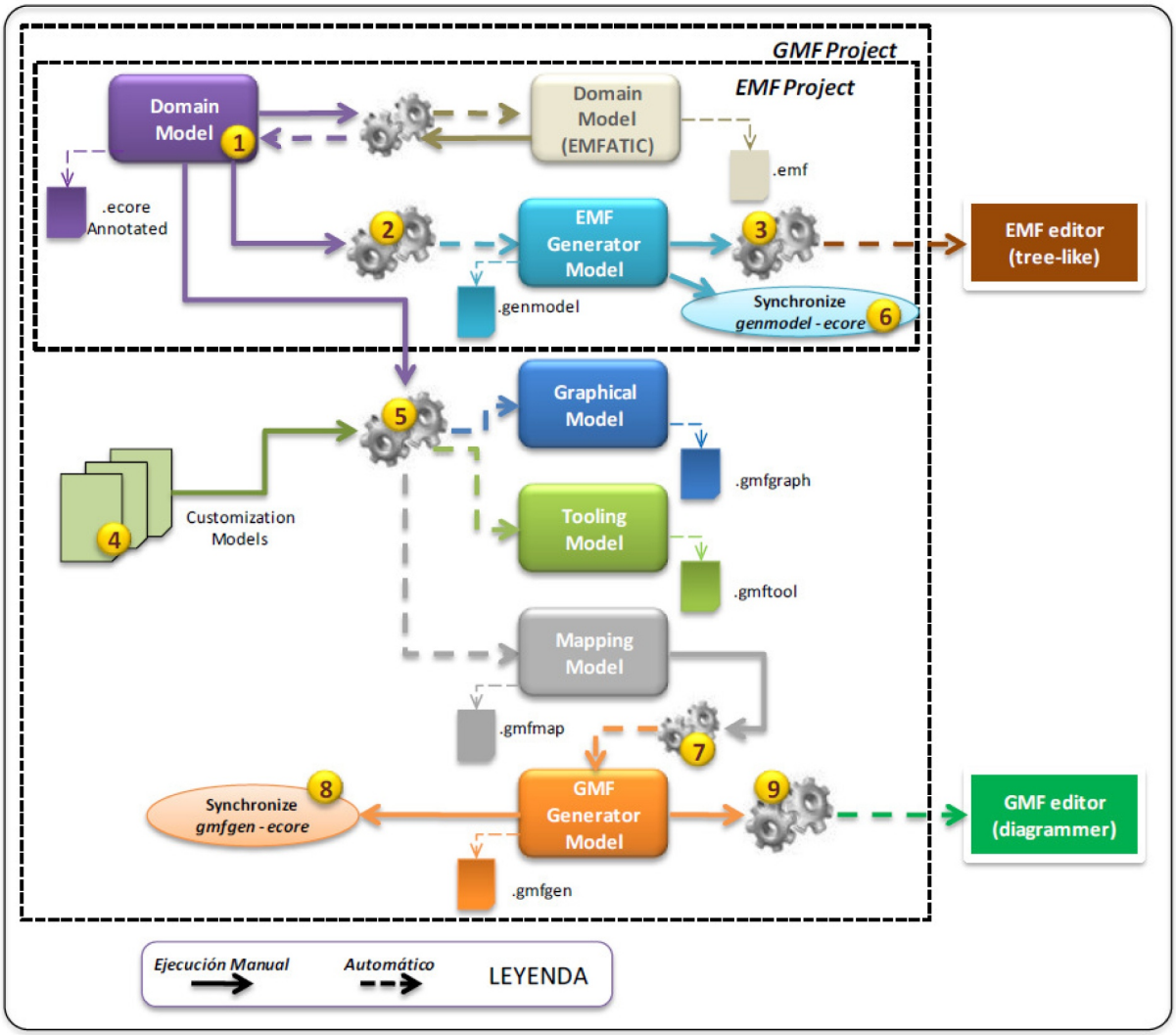

Fig. 1: Proceso de Desarrollo de un editor gráfico usando EuGENia.

3. Anotaciones soportadas

EuGENia es compatible con las siguientes anotaciones realizadas sobre elementos del metamodelo .ecore.

gmf.diagram

Indica el elemento raíz del metamodelo. Sólo puede ser indicado un único elemento Eclass como gmf.diagram. Acepta los siguientes campos:

- diagram.extension: extensión del archivo para el modelo gráfico.
- model.extension: extension del archive para el modelo textual.
- onefile: un valor true indica que el modelo gráfico y textual deben almacenarse en un mismo archivo.

gmf.node

Se aplica a una EClass y denota que debe aparecer en el diagrama como un nodo. Los campos más significativos son los siguientes:

- border: border.color, border.style, border.width. Que permiten indicar el color RGB que se establecerá como color del borde del nodo, el estilo del borde (solido, guiones o puntos), o un entero que especifica el ancho del borde.

- color: indica el color RGB que se establecerá como color de fondo del nodo.
- figure: indica la figura con la que se representará el nodo. Se puede establecer en rectangle, eclipse, rounded, svg, polygon o el nombre completo de una clase Java que implementa la figura.
- label: etiqueta con la que se muestra los valores de los EAttribute o de las EClass, algunos de los campos que soporta son los siguientes: label.icon, label pattern, label.placement, label.readOnly.
- size: indica la dimensión que se utilizará como el tamaño preferido del nodo (por ejemplo 10,5).
- svg.uri: URI del archivo svg que será utilizado como figura.

gmf.link

Indica el elemento que deber representarse como enlace. Puede aplicarse a elementos tipo EClass o elementos tipo EReference. Acepta los siguientes campos:

- color: indica el color RGB del enlace.
- label: El texto estático que se mostrará como etiqueta del enlace.
- source.decoration: La decoración del extremo de origen del enlace.
- target.decoration: Igual que el anterior.
- style: el estilo del enlace
- width: anchura del enlace

gmf.compartment

Define que la referencia de contención creará un compartimento en que los elementos del modelo que se ajustan al tipo de la referencia se pueden colocar.

- collapsible: Se establece en false para evitar que el compartimiento se pueda plegar
- layout: El diseño del layout del compartimento.
- gmf.affixed: Define que la referencia de contención crear nodos que están fijados a los bordes del nodo que lo contiene.

4. Ejemplo de aplicación

```
package ejemplo;
@gmf.diagram(foo="bar")
class Ejemplo {
   val Node[*] nodes;
   val Transition[*] transitions;
}
@gmf.node(label="name",
label.icon="false")
abstract class Node {
   attr String name;
   ref Transition[*]#source outgoing;
   ref Transition[*]#target incoming;
}
@gmf.link(label="name",
source="source", target="target",
target.decoration="arrow")
class Transition {
   attr String name;
   ref Node#outgoing source;
   ref Node#incoming target;
}
```

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto español de investigación EXPLORE (ref. TIN2009-08572) del CICYT y por el proyecto de investigación MISSION-SICUVA (ref. 15374/PI/10) del gobierno de la Región de Murcia.

Referencias

[1] Gronback, R.C. (2009) Eclipse Modeling. Project: A Domain-Specific Language (DSL). Toolkit: Addison-Wesley Professional.

[2] Kolovos, D.S., Paige, R.F., Polack, F.A.C. (2008) The Epsilon Transformation. Language. Theory and Practice of Model Transformations. vol. 5063, 46-60, Springer Berlin/Heidelberg

[3] Eclipse Epsilon Validation Language. http://www.eclipse.org/gmt/epsilon/doc/evl

[4] Kolovos, D.S., Paige, R.F., Polack, F.A.C. (2006) The Epsilon Object Language (EOL). Model Driven Architecture – Foundations and Applications. vol. 4066, Springer Berlin/Heidelberg, pp. 128-142.

[5] Eclipse. Tutorial “Using EMF”. Catherine Griffin, IBM 2002 (revisado en Mayo de 2003).

[6] EuGENia GMF Tutorial http://www.eclipse.org/gmt/epsilon/ doc/articles/eugenia-gmf-tutorial/

[7] IBM. (2004) Emfatic Language for EMF. Development. http://www.alphaworks.ibm.com/ tech/emfatic.