Facilitando el Desarrollo de Software Dirigido por Modelos en Eclipse mediante la Generación Automática de Vistas de Control

Juan F. Inglés-Romero[1], Domingo J. Pérez-Sánchez[1], Cristina Vicente-Chicote[2]
[1] Dpto. Tecnologías de la Información y Comunicaciones, ETSIT, Universidad Politécnica de Cartagena
[2] Dpto. Ingeniería de Sistemas Informáticos y Telemáticos, EPCC, Universidad de Extremadura
Email: juanfran.ingles@upct.es

Resumen. *Este trabajo presenta un entorno de modelado que permite generar automáticamente vistas de control para herramientas de Desarrollo de Software Dirigido por Modelos (DSDM) en Eclipse. Estas vistas muestran, a través de un diagrama de flujo, el proceso que deben seguir los usuarios para llevar a cabo una determinada tarea. El propósito de las vistas generadas es facilitar la integración de varias herramientas relacionadas con la gestión (creación, validación, transformación, etc.) de modelos, mejorando su usabilidad, sobre todo, para usuarios no expertos en los procesos de DSDM con Eclipse.*

1. Introducción

Actualmente, el Desarrollo de Software Dirigido por Modelos (DSDM) [1] es uno de los paradigmas más en boga en el ámbito de la Ingeniería del Software. Los fundamentos sobre los que asienta el DSDM fueron establecidos hace ya un par de décadas. Sin embargo, el auge de este paradigma sólo ha sido posible en los últimos años gracias a la aparición de herramientas que dan soporte a este nuevo enfoque, permitiendo explotar todo su potencial. Entre estas herramientas, cabe destacar Eclipse [2]: una plataforma abierta y de libre distribución que ofrece, entre otras muchas funcionalidades relacionadas con el desarrollo de software, un nutrido grupo de plug-ins relacionados con el DSDM. En los últimos años, Eclipse se ha convertido en el estándar *de facto* para la comunidad de DSDM, ya que implementa las principales tecnologías estandarizadas por el *Object Management Group* (OMG) para dar soporte a este paradigma de desarrollo de software.

Eclipse facilita el desarrollo de herramientas de DSDM, como editores o transformaciones de modelos. Sin embargo, la integración de estas herramientas en un mismo entorno, amigable y sencillo de utilizar para el usuario final, no necesariamente experto en el uso de estas tecnologías, requiere de conocimientos avanzados en programación de plug-ins Eclipse. Esta situación hace que, por lo general, se tengan que dedicar grandes esfuerzos a componer y hacer accesibles en Eclipse las herramientas de DSDM.

Este trabajo presenta un entorno de modelado que permite diseñar y generar, de forma automática, vistas de control para herramientas de DSDM en Eclipse. Estas vistas recogen el conjunto de acciones (p.ej., de creación, validación o transformación de modelos) que deben llevarse a cabo para realizar una determinada tarea, más o menos compleja. Estas acciones son representadas, paso a paso, utilizando diagramas de flujo. Esto facilita y hace más intuitivo el proceso que debe seguir el usuario para completar la tarea.

2. Descripción de la herramienta

Considérese un flujo de modelado para especificar y simular sistemas basados en máquinas de estados jerárquicas, que haga uso de las siguientes tres herramientas de DSDM: (1) un editor de modelos para especificar máquinas de estados jerárquicas; (2) una transformación modelo-a-modelo (M2M) que, dada una máquina de estados jerárquica, devuelva el modelo aplanado equivalente, es decir, sin estados compuestos; y (3) una transformación modelo-a-texto (M2T) que, a partir del modelo generado por la transformación anterior, genere el código asociado a un simulador para dicha máquina de estados, por ejemplo, en Java. Por lo general, herramientas como éstas no suelen desarrollarse de forma integrada, sino como herramientas Eclipse independientes. Su integración, por ejemplo, mediante el uso de menús contextuales que se activen al seleccionar determinados ficheros, requiere conocer en detalle tanto la arquitectura de Eclipse como el proceso necesario para desarrollar nuevos plug-ins que puedan integrarse en dicha arquitectura. Esta tarea puede ser compleja y tediosa, sobre todo para usuarios no experimentados en esta tecnología. A continuación se describen los pasos necesarios para generar automáticamente una vista de control para este ejemplo.

2.1. Modelado de la vista de control

El primer paso consiste en diseñar el diagrama de flujo (flujograma) que se mostrará en la vista. Para ello, se dispone de un entorno gráfico de modelado desarrollado con GMF (*Graphical Modeling Framework*) [3]. En la Figura 1 se muestra el flujograma desarrollado con este editor correspondiente el ejemplo propuesto. El modelo consta de una serie de procesos, asociados a cada una de las transformaciones disponibles (*Flatten FSM* y *Generate Java Code*) y enlazados con los recursos de

entrada y salida según el sentido de las flechas. Un recurso representa uno o más ficheros de tipo: modelo (p.ej. *FSM Model* y *FSM Flat Model*, que se corresponden con el modelo de máquina de estados jerárquica y aplanada, respectivamente) o texto (p.ej. *Java Code*). Además, cada recurso contiene (cero o más) operaciones que pueden ejecutarse sobre él (p.ej., *Select*, para seleccionar un modelo existente; *Validate*, para validar un modelo; o *Edit*, para editar un recurso). El modelo gráfico de la vista de control permite detallar qué elementos la componen, su disposición y apariencia (p.ej., color y estilo), pero no cómo han de actuar los procesos y operaciones incluidas en él.

2.2. Especificación de los procesos y las operaciones

Para especificar los procesos y las operaciones asociadas a cada uno de los recursos diseñados en la vista gráfica anterior se ha desarrollado un lenguaje de modelado textual utilizando el framework Xtext [4]. Se trata de un lenguaje imperativo que incorpora el concepto de documento como tipo de dato primitivo. Un documento puede ser un modelo o un fichero de texto, de forma que el lenguaje incluye primitivas de alto nivel para poder operar sobre cada uno de ellos. Por ejemplo, el método *edit* permite abrir cada documento con el editor adecuado y el método *transform* permite lanzar una transformación, ya sea M2M o M2T. Respecto a las transformaciones de modelos, la versión actual de la herramienta sólo permite ejecutar transformaciones ATL (*ATLAS Transformation Language*) [5] y JET (*Java Emitter Templates*) [6], aunque se prevé la inclusión de otros frameworks de transformación de modelos en el futuro. La Figura 2 muestra un fragmento del código empleado para describir las operaciones del recurso *FSM Model*. Conviene señalar que las signaturas de los diferentes procesos y operaciones se generan automáticamente a partir del modelo gráfico desarrollado en el paso anterior.

2.3. Generación del código de la vista

El último paso consiste en generar el código del plug-in Eclipse asociado a la vista de control. Se trata de una transformación M2T, implementada en JET, que parte del modelo gráfico (detalla la apariencia de la vista) y del modelo textual (especifica procesos y operaciones) para generar el código del plug-in. La Figura 3 muestra el resultado final para el ejemplo propuesto.

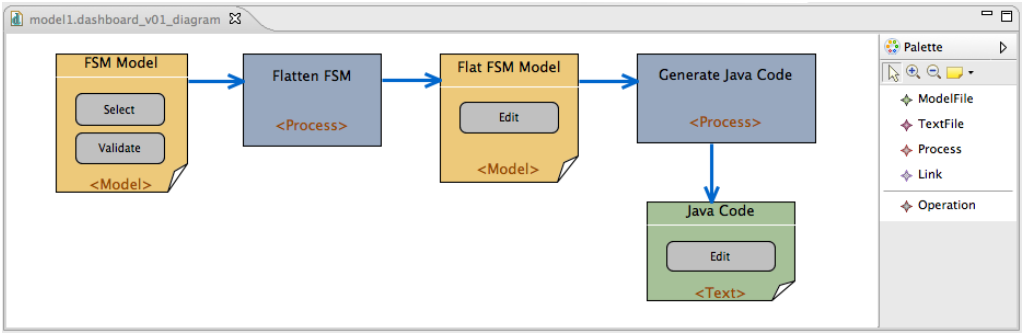

Figura 1. Modelo gráfico de la vista de control.

model1.dashboard

```
modelfile "FSM Model" {
    operator "Select" given
        outputs:
            @FsmModel model<FSM_Metamodel> out1;
    {
        out1 <= open();
    }
    operator "Validate" given
        inputs:
            @FsmModel model<FSM_Metamodel> in1;
    {
        integer result;
        result <= validate(in1);

        if(result == 0) { message("Failed validation"); }
    }
}
```

Figura 2. Fragmento del modelo textual para el ejemplo de las maquinas de estados.

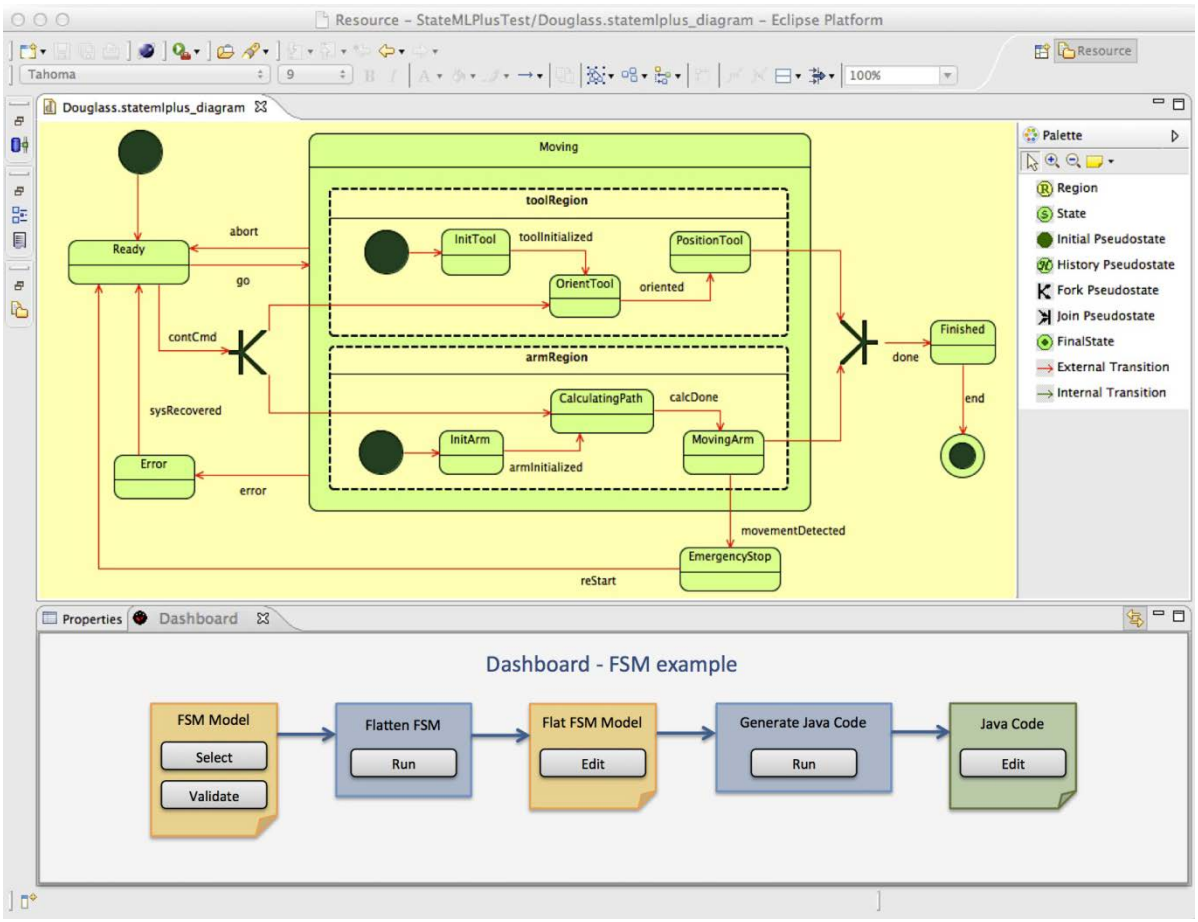

Figura 3. Entorno Eclipse con la vista de control generada.

3. Conclusiones

En este trabajo se ha presentado una herramienta para el modelado y la generación de vistas de control que permiten automatizar procesos de Desarrollo de Software Dirigido por Modelos implementados en Eclipse. Las vistas de control generadas por esta herramienta persiguen dos objetivos: (1) proporcionar a los usuarios un flujograma que, a modo de guía, les indique los pasos que deben seguir para completar una determinada tarea; y (2) facilitar la integración de las herramientas involucradas en el proceso anterior, relacionadas con la gestión (creación, validación, transformación, etc.) de modelos. De este modo, se consigue mejorar la usabilidad de las herramientas y de los procesos dirigidos por modelos, en particular, para aquellos usuarios sin experiencia previa en el manejo de Eclipse.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el MICINN a través del proyecto TIN2011-27430. Juan Francisco Inglés-Romero agradece a la Fundación Séneca su beca de Formación del Personal Investigador (Exp. 15561/FPI/10).

Referencias

[1] Stahl, T., Voelter, M., and Czarnecki, K.: Model-Driven Software Development: Technology, Engineering, Management, ed. Wiley, 2006, ISBN: 978-0-470-02570-3.

[2] Eclipse Modeling Project: http://www.eclipse.org/modeling

[3] GMF - Graphical Modeling Framework: http://www.eclipse.org/modeling/gmp/

[4] Xtext: https://www.eclipse.org/Xtext/

[5] ATL – The ATLAS Transformation Language: https://www.eclipse.org/atl/

[6] JET – Java Emitter Templates: http://www.eclipse.org/modeling/m2t/