**Capitulo 4 - Transiformación de modelo a texto (M2T) con Acceleo**

**4.1 Introducción**

En capitulo anterior se ha visto el metamodelo de contenido (*Content*) para la representación de la sintaxis abstracta de MoWebA con la nueva clasificación de elementos de interfaz, en la cual aparecen a diferencia de MoWebA tradicional que contempla elementos para la web 1.0, elementos enriquecidos que son comunes en las aplicaciones RIA. También junto al metamodelo de contenido, se presentó el metamodelo de posicionamiento (*Layout*), que permite definir la posición de los diferentes elementos del metamodelo de contenido, dentro de una página en particular. Al finalizar la presentación de los metamodelos, se dio pié a los perfiles para el modelado de los *PIM* de la aplicación, extendiendo a UML con las características nuevas propias del DSL MoWebA, específicamente a lo concerniente a contenido y posición.

En este capítulo, describiremos primeramente a la herramienta de transformación de modelo a texto *M2T* *Acceleo*, basado en plantillas, en la cual se presentaran algunas características importantes de la herramienta, como así también, los pasos a seguir para llevar a cabo las transformaciones sobre los PIM de entrada. Seguidamente se presentará a la plantillas para llevar a cabo la transformación de los PIM correspondiente al contenido y a las plantillas de transformación para establecer la configuración de las posiciones de cada uno de los elementos del PIM.

Finalmente se concluirá el capítulo con un ejemplo de transformación M2T para PIM modelados con MoWebA.

**4.2 La herramienta de transformación M2T Acceleo.**

Acceleo es una herramienta de transformación M2T basada en los estándares propuestos por la OMG y que actualmente forma parte de la *Eclipse Foundation*. Acceleo es el resultado de varios años de investigación y desarrollo en el área de los lenguajes de transformación de modelos (MTL) .

**4.2.1 Algunas características importantes de Acceleo**

**Open Source**

Acceleo es un generador de código *open source*. Como tal es posible utilizarlo, bifurcarlo y contribuir con el el.

**Integrado con el Eclipse**

Acceleo está integrado con el *IDE* del *Eclipse*, un editor completo, con corrector de sintaxis, detección de errores en tiempo real, soluciones rápidas, refactorización y mucho más. También contiene vistas dedicadas que ayudan a navegar amigablemente por el generador de código

**Trazabilidad**

Con el generador de código, es fácil perderse en el código generado. De manera a manejar este inconveniente, Acceleo contiene un motor de trazabilidad que permiten encontrar fácilmente que elementos del modelo y que parte del generador (plantilla de transformación) ha sido utilizado para generar la pieza de código.

**Versatilidad**

Generadores de código son a menudo limitados a un conjunto de tecnologías. Con el enfoque basado en templates, Acceleo puede generar código para cualquier tipo de lenguajes. Si es posible escribirlo, Acceleo puede generarlo.

**Generación incremental**

En algún momento podría considerarse adecuado modificar manualmente el código generado por el *template* de transformación y mantener las modificaciones manuales realizadas, en caso que se desea regenerar el código de la aplicación. Acceleo permite definir *protected areas* en al cual se pueden llevar a cabo modificaciones al código generado de manera segura, protegiendo los cambios manuales hecho al código generado.

El lenguaje de plantilla de Acceleo ofrece diversos meta-marcadores que son llamados tags en Acceleo y que son comunes también en otros lenguajes de transfromación M2T.

**Archivos** Para generar código, los archivos deben ser abiertos, llenados y posteriormente cerrados. En Acceleo, existen un tag de archvio especial  *lo cual es utilizado*  para imprimirel contenido que es creado entrel el comienzo y el final del del tag file para un archivo en particular. La ruta y el nombre de archivo ambos son definidos por un atributo del tag.

**Estructura de control:** Existen *tags* para definir estructuras de control tales como *loops (for tag)* para iterar entre colecciones de elementos, por ejemplo, son especialmente útiles para trabajar con referencias multi-valuadas para los resultados navegados en una colección de elementos y en ramas condicionales (if tag).

**Consultas:** Las consultas OCL pueden ser definidas por medio del tag *query*. Los queries pueden ser llamados a travez de todo el template y pueden ser utilizados para factorizar código que es recurrente.

**Expresiones:** Existen expresiones generales para incluir los valores de las expresiones computadas en el texto generado de manera a producir las partes dinámicas del texto de salida. Las expresiones pueden también ser utilizadas para llamar otros *templates* para incluir el código generadas por el *template* llamado en el código producido por *template* llamante. Llamando a otros *templates* puede ser comparado a los métodos en el lenguaje Java.

**Áreas protegidas:** Una característica importante de los lenguajes M2T es el soprote de proyectos donde solamente generación parcial de código es posible. En particular, soporte especial en necesario para guardar código manualmente adherido en el código generado en ejecuciones subsecuentes del generador de código. Para estar tarea, un concepto especial llamado protected areas ha provado ser útil y es exportado por el Acceleo por medio del *tag protected*. Los *protected areas*, son utilizados para marcar secciones en el código generado que no deben ser sobreescritos de nuevo, luego de una nueva ejecución del generador de código. Esta sección típicamente contiene código manualmente escrito.

4.2.2 Entorno de trabajo con Acceleo.

Acceleo propone un ambiente ameno de trabajo basado en el IDE del Eclipse. Uno puede seleccionar la vista propia del Acceleo en el IDE y obtendrá un ambiente personalizado de trabajo con todas las características anteriormente citadas de Acceleo, en donde se podrá ver el editor de plantillas de transformación, la grilla de propiedades, la grilla de errores y la barra exploradora en donde es posible navegar sobre un proyecto el formato de árbol de expansión. En él se encuentran las plantillas de transformación, los modelos de entrada en formato XMI y los módulos de servicio de Java para complementar a las plantillas de transformación. En la figura puede notarse el ambiente de desarrollo con Acceleo.

Para este trabajo de fin de carrera se tuvieron en cuenta las siguientes versiones de herramientas para el proceso de desarrollo con el Acceleo:

Eclipse Kepler Service release 2

Acceleo Versión 3.4

UML Designer for Eclipse Kepler version 3.0

4.2.3 Plantillas de transformación de modelo a texto (Templates).

Las plantillas de transformación tienen un papel preponderante en la generación de código a partir de los modelos de entradas proveídos a Acceleo

4.2.4 Envolvedores de servicios (Service Wrappers).

4.2.5 Operaciones y expresiones OCL en Acceleo.

4.3 Integración de MoWebA con Acceleo.

4.3.1 Introducción.

4.3.2 Transformación de los modelos de MoWebA de MOF a EMF UML2 (v2.x) XMI.

Teniendo en cuenta que el Acceleo solamente le es posible des-serializar modelos de entrada UML en el formato EMF UML 2, es necesario primeramente exportar el proyecto con los modelos PIM y perfiles UML desde la herramienta Magic Draw 16.0 en la cual fueron modelados en primera instancia. Una vez llevado a cabo este paso, el proyecto es importado al Acceleo y de esta forma se tienen los modelos PIM junto a los perfiles UML en la versión UML2 que son los elementos de entrada a la herramienta de transformación, que serán posteriormente des-serializados por medio de las plantillas.

4.6 Generación de código para la interfaz de usuario de MoWebA con Acceleo.

4.6.1 Introducción

4.6.2 Plantilla de transformación para los elementos del perfil de contenido.

Esta plantilla tiene la responsabilidad de llevar a cabo la transformación de los distintos elementos de interfaz definidos en el perfil de contenido. Dentro de los elementos definidos dentro del perfil de contenido tenemos a los elementos que no tienen propiedades enriquecidas y que no tienen características interactivas. Estos elementos son los correspondientes a los de la web 1.0 y son representados por medio de etiquetas y atributos HTML en el cuerpo *(body)*  en una relación uno con la definición del elemento en el modelo y la generación código correspondiente en el tag body del HTML.

Por otro lado se encuentran los elementos con propiedades enriquecidas (RIAS) como los *richToolTip*, *richAccrodion*, *richTabs*, *richDatePicker*, *richAutoSuggest* y los *richFieldLiveValidation* que son parte de la extensión llevada a cabo a MoWebA para este trabajo de fin de carrera. Estos elementos a la par de contar con la sección body del HTML para representar el elemento, también cuentan con una sección *javascript* (encerradas en el *tag script*) para representar la parte dinámica del elemento. La sección correspondiente al tag script contiene el código jQuery correspondiente al elemento definido. Cabe destacar el punto de que el identificador (*id*) de todos los elementos de interfaz, se establecen por medio del nombre del elemento, sin espacios. La identificación de cada uno de los elementos por medio del *id*, resulta importante, principalmente para los elementos de interfaz RIAS, debido a que permiten machear el código *javascript* generado para *jQuery* en la sección del *tag script* (correspondiente a la parte dinámica) con el código HTML generado en el *tag body* para el elemento (correspondiente a la parte estática).

A continuación se describe el proceso de transformación para cada uno de los elementos de interfaz con características de las RIAS

RichToolTip

Cuando la plantilla de transformación encuentra un atributo con el estereotipo *richToolTip* primeramente genera el código jQuery en el *tag script* del *HTML*. A continuación, se genera el código *HTML* en el *tag* *body* para el elemento de interfaz que se encuentra stereotipado junto al *richToolTip,* para resaltar el mensaje que se establece por medio del valor etiquetado *title* que poseen todos los elementos de interfaz en MoWebA.

RichDatePicker

Para este elemento primeramente se genera el código jQuery

4.6.3 Plantilla de transformación para el posicionamiento de los elementos de contenido.

Con la plantilla de transformación para el posicionamiento de los elementos de interfaz dentro de las páginas, es posible generar el código correspondiente a los *cascading style sheets* (css) a partir de los modelos PIM de posicionamiento (Layuout). Primeramente se importan los servicios Java para poder utilizar dentro de la plantilla, expresiones que no son OCL estándar, como por ejemplo el método *hasStereotype* que permite saber si un elemento UML posee cierto estereotipo para llevar a cabo decisiones. Seguidamente se decide el nombre y la extensión del archivo de salida por medio del *tag file* y dentro de este *tag* comienza el proceso de recorrido dentro los elementos del tipo *package*, en donde se busca los valores etiquetados del tipo *cUIElement*. Para cada uno de los valores etiquetados *cUIElement* encontrados dentro de unpaquete estereotipado con *Layout*, se agregan los valores correspondientes a las posiciones definidos en el modelo PIM. Las posiciones a definirse corresponden a los valores en pixeles del *height*, *width*, *xPosition(left)* y *yPosition(top)*



4.7 Conclusiones

Transformaciones de modelo a texto para la generación de interfaces enriquecidas.

En este capítulo se tratará el proceso de transformación de modelo a texto (M2T). El proceso M2T puede tener varios objetivos finales, como podría ser la generación, a partir de los modelos de documentación, código fuente, o cualquier componente de software necesario en una aplicación. Aquí nos concentraremos en la generación de código fuente que representara la interfaz de usuario de una aplicación web representada por medio de modelos que han sido definidos en MoWebA. En primera instancia se presentará lo básico de la generación de código dirigida por modelos, los beneficios de los lenguajes de transformación M2T y luego se presentará el lenguaje de transformación basado en plantillas Acceleo.

Actualmente, la meta central de MDSE es obtener sistemas a partir de los modelos.Las plataformas de ejecución actuales son a menudo basadas en código, con pocas excepciones que permiten una interpretación directa de los modelos. De esta forma, las transformaciones M2T en el area MDSE son am menudo relacionadas con la generación de código para alcanzar la transición a partir nivel del modelo al nivel de código. .

Finalmente se presentarán las plantillas de transformación M2T del Acceleo para los modelos PIM llevado a cabo con MoWebA para la transformación a códgo de los distintos elementos de interfaz y la ubicación de estos dentro de las páginas.

**Conceptos básicos de la generación de código a partir de los modelos**

Mientras que en el contexto de los compiladores, la generación de código, es el proceso de transformar el código fuente en código maquina, en el mundo MDE, la generación de código es el proceso de transformar modelos en código fuente.

Dentro de las preguntas esenciales cuando uno tiene que desarrollar un generador de código basado en modelos se encuentran las siguientes:

**Que tanto va a generarse?:** La pregunta principal aquí es que partes del código puede ser automáticamente generada a partir de los modelos. Es posible llevar a cabo una generación de código parcial o total?. La generación parcial de código puede implicar muchas cosas en este contexto. Primero, puede implicar que una capa (horizontal o vertical) de la aplicaciones completamente generada mientras que otra capa podría ser desarrollada completamente de manera manual. Más aún, también puede implicar que una capa puede ser generada parcialmente y otras partes no cubiertas tienen que ser manualmente completadas. La generación parcial de código, también puede referirse al nivel de modelado, utilizando solamente la generación de código para ciertas partes del modelo, mientras que otras partes no son manipuladas por el generador de código y tiene que ser implementadas manualmente.

**Que va a generarse?** Implica que clase de código fuente va a generarse. Por supuesto, el código a ser generado debe ser lo más conciso posible y debe ser código que puede ser entendido por los desarrolladores. La idea es, generar la menor cantidad de código, que sea capaz de representar un sistema de la mejor manera.

**Como a va generarse?** Muchos lenguajes pueden ser empleados para generar código a partir de los modelos y pueden ser GPLs y DSLs.

**Beneficios de los lenguajes de transformación (M2T)**

**Separación de código estático y dinámico**: Los lenguajes de transformación M2T separan el código estático y dinámico, utilizando el enfoque de *templates* para desarrollar las transformaciones M2T. Una plantilla puede ser vista como una especie de anteproyecto lo cual define elementos de texto estáticocompartidos por todos los actefactos como así también partes dinámicas que deben ser completadas con información especiífica para cada caso en particular. Por lo tanto, un template contiene fragmenteos e extos simple para las partes estáticas y los llamados metamarcadores meta-markers para las partes dinámicas. Los metamarcadores son marcadores de posición y debne ser interpretadaspor un motor de templates que procesa los *templates* y consulta fuentes de datos adicionales para producir las partes dinámicas. Las fuentes adicionales de datos son los modelos. (ver de graficar la figura 9.4)

**Estructura de salida explicita**: Utilizando templates permiten representar explicitamentels estructura del texto de salida dentro del template. Esto permite una especificación de la generación de código más entendible y leíble.

**Lenguaje de consulta declarativa**: dentro de los metamarcadores, código es utilizado para acceder a la información almacenada dentro de los modelos. OCL, es la elección para llevar a cabo esta tarea en la mayoría de los lenguajes de transformación. De esta forma, los lenguajes de transformación M2T también permiten el uso de OCL ) o dialecto de OCL) para especificar a los metamarcadores.

**Funcionalidad base reusable**: Los lenguajes de transformación M2T actuales, vienen con soporte de herramienta, lo cual permite leer directamente los modelos y serialiazar texto en archivos, definiendo solamente archivos de configuración.

**Una vista de los lenguajes de transformación basados en templates**

Diferentes lenguajes basados en templates existen los cuales pueden ser empleados para generar texto a partir de los modelos.

**XSLT**: La serialización XMI de los módulos pueden ser procesados con XSLT, que es el estándar W3C para transformar documentos XML en documentos arbitrarios de texto. Sin embargo, en este caso, los scripts de generación de código tiene que ser implementados basados en la serializaición XMI que requiere ciertos conocimientos adicionales de como los modelos son actualmente codificados como archivos XML. Así, el enfoque opera directamente a nivel de modelo.

**JET Java Emitter Template** fue uno de los primeros enfoques para de desarrollo del EMF para la ón de código basado en modelos. Pero Jet no está limitada a modelos EMF. En general, con JET, todo objeto basado en Java es transformable a texto. JET provee una sintaxis similar a JSP adaptada a la estructura template para transformación M2T. Expresiones Java arbitrarias puede ser incrustadas en los *templates* JET. Más aún, los *template* de JET son transformados a código Java puro para propósitos de ejecución. Sin embargo, no tiene un lenguaje de consulta dedicado para los modelos disponibles en JET.

**Xpand**: Este lenguaje de transformación provee un lenguaje dedicado para consultar modelos siendo este muna mixtura de Java y OCL (especialmente muchos iteradores basados en OCL están disponibles). La continuación a este proyecto se llama Xtend, que está basado en *Java* pero ofrece muchas características adicionales en el lenguaje. Por ejemplo, es posible incrustar *templates* de generación de código (para tener una sintaxis similar al template Xpand) dentro del código Xtend.

**MOFScript**: Este proyecto provee otro lenguaje de transformación M2T proveyendo características similares tales como Xpand. MOFScript ha sido desarrollado como una propuesta de estandarización para la OMG y se encuentra disponible como un *plug-in* para el Eclipse y soporta modelos del tipo EMF.