**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 iterrar 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 expreseiones 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 utilizadoas para llamar otros templates para inclir el código generadopaor el template llamado en el código producido por template llamante. Llamando a otros templates puede ser comparado a los métodos en el lenguaje Java.

Areas 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 manuelamente aderido 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 sportado por el Acceleo por medio del tag protected. Los protected areas, son utilizado 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 manuelmente 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