Automatización de la Selección de Transformaciones Alternativas Basada en Atributos de Calidad

Javier Gonzalez-Huerta
Grupo de Investigación ISSI
Depto. de Sistemas Informáticos y Computación.
Universidad Politécnica de Valencia
46022 Valencia.
jagonzalez@dsic.upv.es

Emilio Insfran
Grupo de Investigación ISSI
Depto. de Sistemas Informáticos y Computación.
Universidad Politécnica de Valencia
46022 Valencia.
einsfran@dsic.upv.es

Silvia Abrahão
Grupo de Investigación ISSI
Depto. de Sistemas Informáticos y Computación.
Universidad Politécnica de Valencia
46022 Valencia.
sabrahao@dsic.upv.es

Resumen

Las transformaciones de modelos son una pieza clave en el Desarrollo de Software Dirigido por Modelos. Una transformación de modelos es el proceso por el cual un modelo puede convertirse en otro modelo del mismo sistema, usando un lenguaje para describir dicha transformación. Dado un modelo origen se pueden obtener uno o varios modelos destino atendiendo a las transformaciones alternativas seleccionadas. Las transformaciones alternativas aparecen cuando una entidad del modelo origen puede ser transformado en más de una entidad distinta en el modelo destino. Estos modelos, aun siendo similares en cuanto a la realidad que representan, pueden ser muy diferentes con respecto a sus atributos de calidad. En este artículo se define cómo introducir atributos de calidad en los procesos de transformación con el fin de dirigir la selección de transformaciones alternativas. Finalmente se muestra una estrategia de automatización para transformar diagramas de secuencia de un modelo de requerimientos en diagramas de clase UML.

Palabras clave: Transformación de modelos, Desarrollo de Software Dirigido por Modelos.

1 Introducción

En los últimos años el Desarrollo de Software Dirigido por Modelos (DSDM) [15] ha ganado gran aceptación como una aproximación para el desarrollo de sistemas en el que los modelos pasan a ser artefactos de primer orden. El uso de modelos eleva el nivel de abstracción, posibilitando un incremento de la productividad en el proceso de desarrollo de software y de la calidad del código generado a partir de los modelos [16]. Una de las ideas clave en el DSDM es transformar modelos con un alto nivel de abstracción en modelos más concretos que serán finalmente transformados en el código final. Por tanto, la calidad de los modelos involucrados en el proceso de desarrollo pasa a ser un factor decisivo para obtener software de calidad.

Una transformación es el proceso por el cual un modelo puede convertirse en otro modelo del mismo sistema, usando un lenguaje para describir dicha transformación [10]. En un proceso de transformación de modelos, un modelo origen puede ser transformado en uno o varios modelos destino, que aun siendo similares en cuanto a la realidad que representan, pueden ser muy diferentes con respecto a sus atributos de calidad. Para poder producir artefactos software con unos determinados atributos de calidad, es necesaria la identificación de transformaciones alternativas y el análisis de cómo la aplicación de esas transformaciones afecta a la calidad de los artefactos obtenidos [9]. A pesar de que en la propia definición de MDA [10], OMG sugiere que se podrán utilizar una amplia gama de requisitos de calidad de servicio para dirigir transformaciones, no se encuentran aproximaciones que introduzcan, de manera genérica y efectiva, los requisitos de calidad como factor de decisión a la hora de seleccionar entre transformaciones alternativas.

En un trabajo previo se mostró una arquitectura genérica para dar soporte a las transformaciones de modelos dirigidas por atributos de calidad [6], presentando un conjunto de artefactos para representar la información adicional necesaria (un modelo de calidad y un modelo de transformaciones) y se definía un proceso para dar soporte a estas transformaciones.

En este trabajo, haciendo uso de dicha arquitectura, definimos un esquema para, partiendo de una transformación de modelos existente, automatizar la selección de transformaciones alternativas en base a los atributos de calidad que ha de tener el modelo destino.

La organización de este trabajo es la siguiente: la sección 2 describe los trabajos relacionados. La sección 3 presenta una aproximación para la automatización de la selección de transformaciones alternativas. La sección 4 presenta una implementación del esquema de automatización propuesto a un dominio concreto. Finalmente, la sección 5 presenta las conclusiones y trabajos futuros.

2 Trabajos Relacionados

En esta sección, analizamos algunas propuestas que abordan la problemática de la selección de transformaciones alternativas en procesos de transformación de modelos desde la perspectiva de algún atributo de calidad.

Kurtev [9] propone una técnica formal para la definición de espacios de transformación que soporta el análisis de las transformaciones alternativas dado un modelo origen. Esta técnica ofrece operaciones de selección y reducción de los espacios de transformación basados en ciertas propiedades de calidad que se desea que tenga el modelo destino resultante del proceso de transformación. Para generar el espacio de transformación, se define un proceso que toma como entrada el modelo origen y su metamodelo, el modelo destino y un Modelo de Calidad. Los atributos de calidad son empleados como criterio de selección entre las distintas alternativas y son incluidas explícitamente en los espacios de transformación. Concretamente esta propuesta explora la adaptabilidad y reusabilidad de las transformaciones de modelos para obtener esquemas XML a partir de modelos de clase. Para lograr la automatización el autor desarrolla también su propio lenguaje de transformación llamado MISTRAL apoyándose en una aproximación para el procesamiento de XML basado en un framework de modelado basado en MOF.

Merilinna [11] propone una herramienta para guiar, de manera automatizada, las transformaciones de modelos de arquitecturas software, basándose en criterios de calidad. La aproximación considera únicamente las transformaciones horizontales [13] (transformaciones PIM a PIM) que son aquellas que se ejecutan al mismo nivel de abstracción. Si aparecen múltiples alternativas, el arquitecto determina qué arquitectura es la mejor solución teniendo en cuenta la información recogida en una base de datos denominada *StyleBase*. Esta base de datos contiene un conjunto de patrones de diseño y estilos arquitectónicos. Para cada patrón se almacenan los atributos que maximiza. Los autores definen su propio lenguaje para dar soporte a las transformaciones Q-RDL (*Quality-Driven Rule Definition*) y llevan a cabo una implementación de una herramienta Q-TRA desarrollada en C++.

En definitiva, ambas propuestas definen una infraestructura propietaria, definiendo su propio lenguaje de transformación (MISTRAL y Q-RDL). Nuestra propuesta se ha apoyado en herramientas abiertas y extensibles, haciendo uso de facilidades del proyecto Eclipse como EMF (*Eclipse Modeling Framework*) [3] y de lenguajes de transformación estándar como QVT-Relations [12]. Además ambas propuestas [9], [11], y otras que abordan esta problemática, cubren parcialmente el ciclo de vida del DSDM soportando únicamente transformaciones horizontales o verticales. Por el contrario, nuestra propuesta puede ser utilizada en cualquier dominio, definiendo adecuadamente los artefactos necesarios, y en los distintos procesos de transformación que aparecen en las distintas fases del ciclo de vida DSDM. Además permite trabajar con múltiples atributos de calidad de manera simultánea efectuando un análisis trade-off para ponderar la importancia relativa de cada uno de los atributos de calidad.

3 Transformación de modelos dirigida por atributos de calidad

En esta sección se introducen un esquema de automatización para la selección de transformaciones alternativas en base a atributos de calidad empleando la arquitectura propuesta en [6]. La sección 3.1 presenta una visión general de una arquitectura para transformaciones de modelos dirigidas por atributos de calidad, y la sección 3.2 define la estructura de las reglas de

transformación para adaptarlas al proceso de transformación de modelos dirigido por atributos de calidad empleando haciendo uso de multimodelos. Un multimodelo es una colección de modelos que dan soporte a diferentes vistas de un sistema, caracterizados por la existencia de relaciones entre los elementos de sus correspondientes metamodelos [4].

3.1 Una arquitectura para transformaciones de modelos dirigidas por atributos de calidad

Para dar un soporte adecuado a la definición de transformaciones de modelos que tengan en cuenta atributos de calidad, se ha propuesto una arquitectura basada en transformaciones con multimodelos.

A diferencia de las transformaciones convencionales, los modelos participantes en transformaciones con multimodelos están relacionados entre sí. En el caso del proceso de transformación de modelos dirigido por atributos de calidad se emplean tres modelos adicionales, dos de ellos de entrada y uno intermedio generado por el propio proceso.

El primer modelo de entrada es el *Modelo de Calidad*. Permite al usuario de la transformación seleccionar los atributos de calidad que ha de tener el modelo destino. El *Modelo de Calidad* tiene una estructura jerárquica basada en el estándar ISO/IEC 25000 SQuaRE [8], con características, sub-características y atributos de calidad.

El segundo modelo de entrada es el *Modelo de Transformaciones* que expresa las relaciones entre los atributos de calidad y las transformaciones alternativas. Los atributos de calidad que en el aparecen son un subconjunto de los que aparecen en el *Modelo de Calidad* y han sido identificados como relevantes por el experto del dominio.

El tercer modelo, generado por el propio proceso es el *Modelo de Reglas Activas*, que representa, para un determinado conjunto de constructores en el modelo origen, que reglas de transformación han sido seleccionadas de entre las alternativas para llevar a cabo la transformación.

La Figura 1 presenta la estructura general del proceso de transformación dirigido por atributos de calidad mostrando los artefactos, actividades y fases principales.

En la primera fase, el experto de dominio lleva a cabo la *Identificación de Transformaciones Alternativas* que serán posteriormente adaptadas al proceso de transformación de modelos dirigido por atributos de calidad. El experto del dominio lleva a cabo además un *Análisis Trade-Off* entre los atributos de calidad y las diferentes transformaciones alternativas. El objetivo es establecer cómo la aplicación de las distintas transformaciones alternativas afectará a la calidad del modelo destino. Esto se puede llevar a cabo mediante el uso de evidencia empírica obtenida en experimentos controlados o con el conocimiento del experto del dominio.

Con el fin de realizar el *Análisis Trade-Off* en el que se medirá por un lado la importancia relativa de cada uno de los atributos de calidad, y por otro el grado con que las transformaciones alternativas dan soporte a los atributos de calidad empleamos el Proceso de Análisis Jerárquico (AHP) [14]. El AHP es una técnica de toma de decisiones que ha sido frecuentemente utilizada para resolver conflictos de esta índole. Este proceso se realiza i) estudiando cada patrón estructural, definido como un conjunto de constructores del metamodelo origen, para el que se han identificado transformaciones alternativas, y ii) analizando cómo la aplicación de cada una de las alternativas afecta a los distintos atributos de calidad.

Los resultados del *Análisis Trade-Off* son introducidos en el *Modelo de Transformaciones*, que contiene los atributos de calidad identificados como relevantes por el experto del dominio, los pesos de cada atributo de calidad (entendidos como la importancia relativa de cada uno de estos atributos de calidad) y la información de los patrones estructurales para los cuales se han podido identificar transformaciones alternativas. Para cada patrón estructural para el que han sido identificadas transformaciones alternativas, se asociará la información de cada una de las transformaciones alternativas: el conjunto de reglas de transformación asociadas, y el impacto que tiene la aplicación de cada alternativa sobre los atributos de calidad. Estos impactos son flexibles y podrán ser modificados con posterioridad si se constata que la aplicación de alguna de las alternativas no tiene el impacto esperado en los atributos de calidad de los modelos generados.

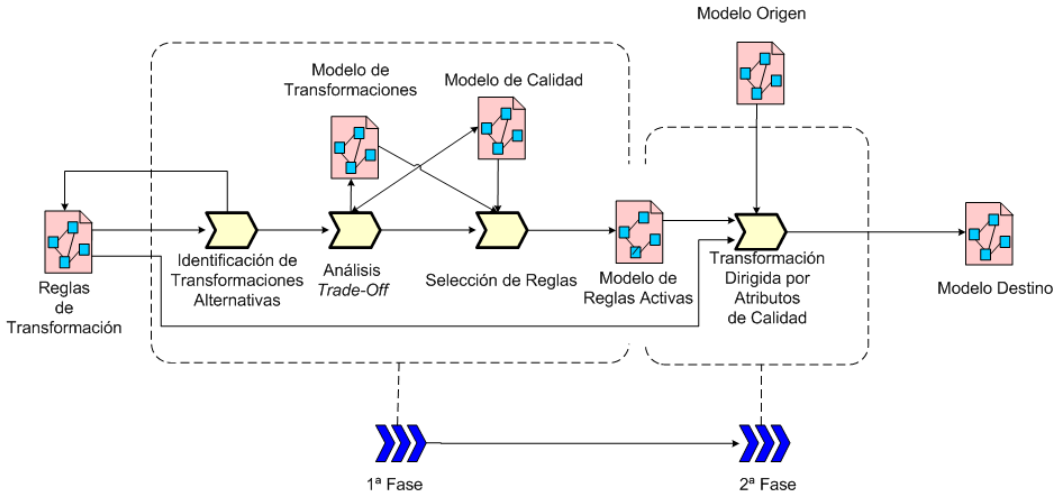

Figura 1. Fases y artefactos involucrados en el proceso de transformación

A continuación, cada vez que el usuario cambie los atributos de calidad que ha de tener el modelo destino, se lleva a cabo la *Selección de Reglas,* que tomando como entrada el *Modelo de Transformaciones* y el *Modelo de Calidad*, en el que el usuario de la transformación ha seleccionado los atributos que ha de tener el modelo resultante, ejecuta un proceso de transformación que automáticamente genera el *Modelo de Reglas Activas*.

En la segunda fase se utiliza el *Modelo Origen* y el *Modelo Reglas Activas* como entrada para generar el *Modelo Destino* (ver Figura 1) además de las reglas de transformación, con aquellas reglas que no son alternativas y las que son alternativas que serán discriminadas por el *Modelo de Reglas Activas*.

De esta forma cuando en el *Modelo Origen* se encuentra un patrón estructural para el que se han identificado transformaciones alternativas se llevará a cabo la transformación haciendo uso de las reglas asociadas a este patrón en el *Modelo de Reglas Activas*. Las reglas de transformación aplicadas en esta fase aseguran que los atributos de calidad seleccionados sobre el *Modelo de Calidad* durante la fase 1 estén presentes en el Modelo Destino.

3.2 Definición de la estructura de las reglas de transformación

Esta sección presenta la estructura de las reglas de transformación para adaptarlas al proceso de transformación de modelos dirigido por atributos de calidad y lograr con ello la automatización. Para ello, nos apoyaremos en conceptos del lenguaje de transformación QVT-Relations [12], aunque los conceptos definidos en esta sección y el proceso de transformación propuesto puedan ser aplicados a otros marcos tecnológicos y lenguajes de transformación de modelos.

El punto de partida de este proceso es una transformación de modelos existente en la que existen patrones estructurales para los que se han identificado transformaciones alternativas.

Una vez se han identificado el conjunto de reglas que representan transformaciones alternativas, se lleva a cabo una refactorización de las reglas de transformación, para conseguir reglas definidas sobre patrones estructurales con un tamaño homogéneo y lo más pequeño posible, con el fin de maximizar la cohesión y minimizar el acoplamiento.

Las reglas de transformación se descomponen en dos niveles de reglas. Las reglas *top* se activan

cada vez que se encuentra el patrón estructural en el modelo de origen y las reglas *non-top* son invocadas directa o transitivamente desde la cláusula *where* de las reglas *top*.

Las reglas de transformación incluidas en las transformaciones alternativas han de ser modificadas para adaptarlas a la estructura de la arquitectura. Por cada patrón estructural para el que se han identificado transformaciones alternativas generaremos una nueva regla *top* por composición de todas las alternativas, aplicando principios de composición análogos a los descritos en [2] para el contexto de transformaciones QVT-Operational. Esta regla tendrá discriminantes que le permitirán seleccionar apropiadamente la alternativa, y realizar las invocaciones a reglas *non-top* en función de las selecciones realizadas en la fase de análisis de reglas. La información de las distintas reglas a invocar se encuentra disponible en el *Modelo de Reglas Activas*.

En QVT-Relations una transformación se define como una relación entre dos o más modelos. Una relación en una transformación define las restricciones que deben ser satisfechas por los elementos de los modelos participantes. Una relación se define por dos o más dominios y un par de predicados *when* y *where* que especifican la relación que debe mantener entre los elementos de los modelos participantes. La cláusula *when* especifica las condiciones bajo las cuales la relación debe ejecutarse. La cláusula *where* especifica la condición que debe ser satisfecha por todos los elementos del modelo que participan en la relación. Estas cláusulas pueden restringir alguna de las variables de la relación y sus dominios [12].

La estructura de una regla *top,* que agrupa las distintas alternativas, constará de tres dominios. El primer dominio es el *Modelo Origen*, donde son descritas con precisión las entidades del patrón estructural. El segundo dominio es el *Modelo de Reglas Activas*, donde se define la asociación entre el nombre de las alternativas para un patrón estructural dado y las distintas reglas *non-top* que crearán o modificarán los constructores del modelo destino. Por último en una regla *top* tendremos también el dominio del *Modelo Destino* donde se define aquellos elementos comunes que serán creados o modificados por las diferentes alternativas. La precondición de la regla *top* consta de restricciones sobre el dominio origen y sobre el *Modelo de Reglas Activas*, así como otras condiciones que limiten la ejecución de la regla. La postcondición de las reglas *top* son las llamadas, por composición de transformaciones [2], a las distintas reglas *non-top,* agregando un mecanismo de reutilización a las reglas *top*.

En cuanto a las reglas *non-top*, su estructura será definida de la forma habitual para representar la relación entre patrones estructurales del modelo origen y sus correspondencias en el modelo destino.

4 Automatizando la selección de transformaciones alternativas

Esta sección muestra la automatización del proceso de transformación de modelos dirigido por atributos de calidad en un dominio concreto. Por restricción de espacio, presentamos únicamente un breve resumen de la automatización de la transformación de un modelo de requisitos, en concreto de diagramas de secuencia a diagramas de clase UML. Esta implementación se ha llevado a cabo con la infraestructura Eclipse, empleando EMF para definir los metamodelos y la persistencia en XMI que brinda para serializar los modelos. Como lenguaje de transformación empleamos QVT-Relations y como motor de transformación empleamos MediniQVT.

4.1 Identificación de alternativas y atributos relevantes

El proceso comienza con la identificación de transformaciones alternativas en el dominio específico. En este caso utilizamos el Catalogo de Reglas de Transformación propuesto por Insfran [7]. Este catálogo define las relaciones entre patrones estructurales del diagrama de secuencia y el diagrama de clases UML entre otros. En concreto analizando estas reglas de transformación mostramos como ejemplo un patrón estructural en el diagrama de secuencia que da lugar a más de una transformación alternativa para el diagrama de clases UML. Este patrón estructural se muestra en la Figura 2: "un mensaje con el estereotipo «*Service/New*» de la clase A a la B, y dos mensajes con el estereotipo «*Connect*» de la clase B a las clases C y D,".

A continuación se muestran algunas de las transformaciones que pueden aplicarse para este patrón estructural:

- *Alternativa 1:* aplicar dos veces la regla de transformación TR15 del catálogo de reglas de transformación que transforma un mensaje con el estereotipo «*Connect*» en una relación de asociación entre las clases [7].
- *Alternativa 2:* Aplicar la regla de transformación TR39 que transforma un mensaje con el estereotipo «*Service/New*» de la clase A a la clase B, y dos mensajes con el estereotipo «*Connect*» de la clase B a las clases C y D, respectivamente, en una nueva clase asociación (B) y una relación de asociación entre las clases C y D [7].

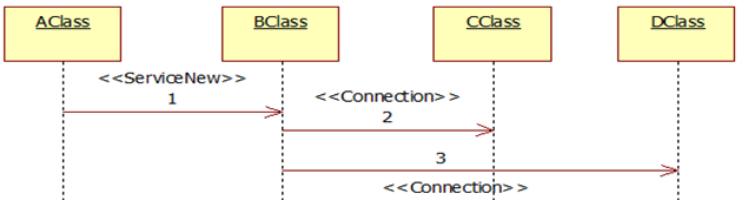

Figura 2. Patrón estructural del modelo origen

Las diferentes transformaciones alternativas se definen mediante la creación de nuevas reglas top que agrupan las reglas alternativas para un determinado patrón estructural. Estas reglas top contienen la referencias al *Modelo de Reglas Activas*. Con la información que proviene del *Modelo de Reglas Activas*, el patrón se transforma por medio de llamadas a un conjunto de reglas *non-top* u otro.

La regla *non-top* llamada desde la alternativa 1, crea una asociación entre dos clases. Esta regla será llamada dos veces, una con los argumentos B y C y otra con B y D. La regla *non-top* llamada desde la alternativa 2, crea una asociación entre dos clases y un enlace a clase asociación con una tercera clase. Esta regla será llamada una vez con los argumentos C, D y B.

Se han identificado otros patrones: dos clases conectadas por un mensaje con el estereotipo *«Service/New»* o *«Connect»* que serán transformadas en asociaciones entre las clases, y para las que no identificamos transformaciones alternativas. Para la transformación de estos dos patrones definimos únicamente las reglas *top* y reutilizaremos las llamadas a las reglas *non-top* ya definidas para convertir ese patrón en una asociación entre las clases. En la precondición de la regla, habrá una llamada a una consulta para evitar colisiones entre reglas.

Para el ejemplo que nos ocupa, se han seleccionado dos características de calidad que deben tener los modelos generados por el proceso de transformación: operabilidad y mantenibilidad. En concreto, se utilizó una sub-característica de la operabilidad: *Appropriateness Recognizability* y la sub-característica de mantenibilidad: *Changeability* tal como se define en el nuevo estándar ISO/IEC 25000 SQuaRE [8]. Los atributos seleccionados serán, en ambos casos, la efectividad y la eficiencia de cada una de estas subcaracteristicas. En la actividad de *trade-off* el experto del dominio asigna pesos a cada una de las alternativas en relación a estos atributos, basándose en su propia experiencia o bien en evidencias empíricas obtenidas en experimentos controlados en los que se demuestre la relación entre las transformaciones alternativas y los atributos de calidad.

La tabla 1 muestra los distintos pesos asignados en la fase de *Trade-Off*, la base de estos valores se basa en los resultados de dos experimentos controlados [1], [5]. Dichos valores se basan en, por un lado, que la presencia de clases asociación en un modelo de clases UML empeora la *Appropriateness Recognizability* con respecto a modelos de clases UML que contienen asociaciones [1] y, por otro lado, en el hecho de que el incremento del número de asociaciones en un modelo de clases UML empeora la mantenibilidad del modelo [5]. La alternativa 1 tiene mayor número de asociaciones (2 por cada aplicación) que la alternativa 2 (1 por cada aplicación).

La justificación de la asignación de los pesos a cada una de las alternativas se trata con mayor detalle en [6].

Tabla 1. Pesos de las diferentes alternativas

	Appropiatness Recognizability		Changeability	
	Efic.	Efec.	Efic.	Efec.
Alternativa 1 (TR15)	0.40	0.35	0.25	0.14
Alternativa 2 (TR39)	0.36	0.30	0.40	0.37

4.2 Aplicación de las transformaciones

Esta sección muestra como un Diagrama de Secuencia puede generar diferentes diagramas de clase UML en función de la selección realizada sobre el *Modelo de Calidad* por el usuario de la transformación. Para ello, utilizamos como modelo de origen un diagrama de secuencia de la especificación de un Sistema de Gestión Hotelera. Este diagrama de secuencia muestra la interacción entre los objetos para la especificación del caso de uso de alquiler de habitaciones, que es iniciado por el actor *employee*. Este caso de uso representa el alquiler de una habitación por un cliente. Por razones de brevedad, se presenta sólo uno de los diagramas de secuencia, mostrado en la Figura 3.a.

En la actividad de *Selección de Reglas*, se ejecuta una transformación que, dependiendo de las selecciones hechas por el usuario sobre el *Modelo de Calidad*, tal como se muestra en la Figura 3.b, genera un *Modelo de Reglas Activas*. En este modelo, para el patrón estructural descrito en la Figura 2, se habrá seleccionado la regla de transformación alternativa que corresponde.

En la actividad de *Transformación Dirigida por Atributos de Calidad* se aplica este *Modelo de Reglas Activas* para la generación de los posibles modelos destino mostrados en la Figura 4.a y 4.b.

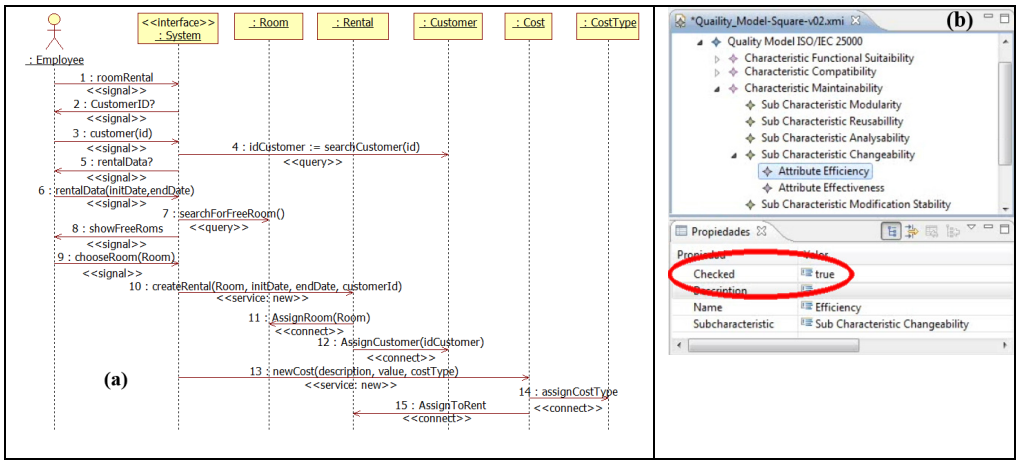

Figura 3. Modelos de entrada a la fase de transformación

La selección por parte del usuario del atributo de calidad *Changeability* tal como se muestra en el *Modelo de Calidad* mostrado en la Figura 3.b da como resultado el diagrama de clase UML mostrado en la Figura 4.b en el que el patrón estructural formado por los mensajes 10 al 12 y el patrón estructural formado por los mensajes 13 al 15 del modelo origen se transforman en dos clases asociación. La selección del atributo de calidad *Appropriateness Recognizability* da como resultado el diagrama mostrado en la Figura 4.a en el que el patrón estructural formado por los mensajes 10 al 12 y el patrón estructural formado por los mensajes 13 al 15 en el modelo origen se transforman en relaciones de asociación entre las distintas clases. Las dos representaciones que se muestran para cada modelo en la Figura 4.a y b son, el modelo destino tal como es generado por la herramienta de transformación y su representación como diagramas de clase UML.

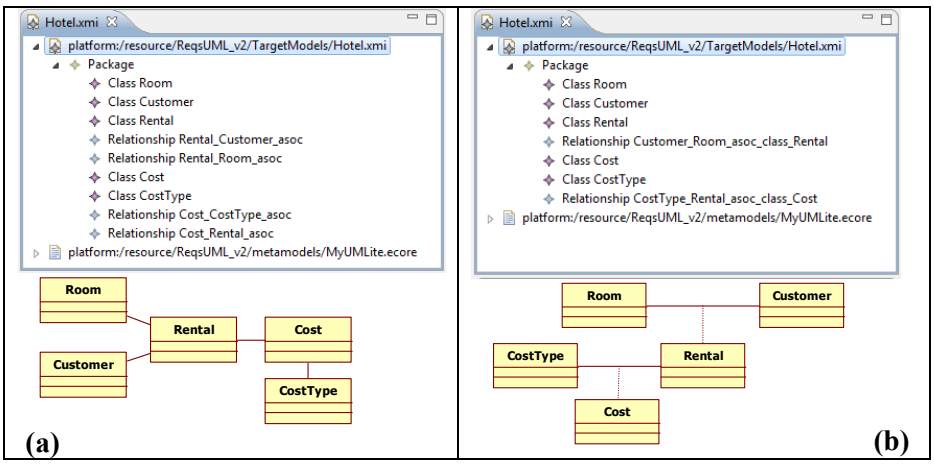

Figura 4. Modelos destino alternativos

En esta sección se ha mostrado una implementación de la arquitectura a un contexto de transformaciones en los que se presentan alternativas, donde se puede observar el impacto de la selección de los atributos de calidad en el modelo de destino. Todos los conceptos descritos en este ejemplo pueden ser aplicados a transformaciones más complejas, bien sea por el tamaño de los modelos de origen y/o destino, como por el número de transformaciones alternativas identificadas.

5 Conclusiones y trabajos futuros

Este artículo ha presentado la automatización de un proceso de selección de transformaciones alternativas basándonos en atributos de calidad en transformaciones de modelos. Se ha descrito el enfoque multimodelo para la definición de transformaciones donde aparecen alternativas, así como la estructura que han de tener las reglas de transformación. Este enfoque permite la automatización de las transformaciones, según la información de los atributos de calidad deseados. Si bien no se está analizando la calidad global durante el proceso de transformación sino como la aplicación de cada transformación impacta en la calidad del modelo generado, el enfoque global analizando la evolución de la calidad del modelo generado será abordado en trabajos futuros.

La viabilidad del método se ilustra con un ejemplo en el que se automatiza la selección de transformaciones alternativas en la transformación de diagramas de secuencia en diagramas de clase UML. También se muestra cómo la selección de cada alternativa afecta a la calidad de los modelos destino generados, mostrando ejemplos generados de forma automática por la herramienta de transformación. La adopción de este enfoque puede extender los procesos de desarrollo de software dirigido por modelos actuales brindando directrices para la definición de procesos en los que aparecen transformaciones alternativas, para que la selección entre estas se lleve a cabo en base a atributos de calidad.

Como trabajos futuros, se pretende aplicar este enfoque a otros dominios y poner a prueba el proceso de transformación con múltiples atributos de calidad. Se pretende definir métricas que evalúen aspectos estructurales de los modelos generales para medir la calidad de los modelos destino generados, complementándolas con valoraciones subjetivas de expertos que servirán para *retroalimentar l*a actividad de trade-off. Se pretende además diseñar experimentos controlados que validen empíricamente la usabilidad y efectividad de la propuesta. También planteamos abordar un análisis de la calidad global del modelo generado, analizando el impacto que tiene la secuencia de aplicación de reglas y las posibles incompatibilidades que puedan aparecer.

Agradecimientos

Este trabajo ha sido cofinanciado por el Ministerio de Ciencia e Innovación, en el marco del proyecto MULTIPLE (ref. TIN2009-13838), y por la G.V-A (proyecto CALIMO con ref. GV/2009/103).

Referencias

[1] Abrahão, S., Genero, M., Insfrán, E., Carsí, J. A., Ramos, I., & Piattini, M. Quality-Driven Model Transformations: From Requirements to UML Class Diagrams. In J. R. (Eds.), Model-Driven Software Development: Integrating Quality Assurance. IGI Global Publishing, USA, 2008.

[2] Belaunde, M. Transformation Composition in QVT. First European Workshop on Composition of Model Transformations (CMT). Bilbao, Spain, 2006.

[3] Budinsky, F., Steinberg, D., Ellersick, R., Merks, E., Brodsky, S.A., Grose, T.J. Eclipse Modeling Framework. Addison Wesley, 2003.

[4] Boronat, A., Knapp, A., Meseguer, J. Wirsing, M. What is a Multi-Modeling Languaje? In: LNCS, vol 5486/2009, pp 71—87 Springer, Heidelberg, 2009.

[5] Genero, M., Manso, M. Esperanza, Visaggio, C. A., Canfora, G., Piattini, M. Building measure-based prediction models for UML class models maintainability. Empirical Software Engineering 12(5): 517-549, 2007.

[6] Gonzalez-Huerta, J., Blanes, D., Insfran, E., Abrahão, S. Towards an Architecture for Ensuring Product Quality in Model-Driven Software Development. Procedings of the International Conference on Product Focused Software Development and Process Improvement (PROFES 2010) Limerick, In press, 2010.

[7] Insfrán, E. A Requirements Engineering Approach for Object-Oriented Conceptual Modeling. Phd Thesis, Universidad Politécnica de Valencia, Valencia, 2003.

[8] International Organization for Standardization ISO/IE Software engineering 2005 Software product Quality Requirements and Evaluation (SQuaRE), 2005.

[9] Kurtev, I. Adaptability of Model Transformations. PhD Thesis. University of Twente, Twente, 2005.

[10] MDA Guide Version 1.0.1. http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf, 2003.

[11] Merilinna, J. A Tool for Quality-Driven Architecture Model Transformation. PhD thesis. VTT Technical Research Centre of Finland, Finland, 2005.

[12] Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification - http://www.omg.org/spec/QVT/1.0/PDF, 2006.

[13] Mens, T., Van Gorp, P. A Taxonomy of model Transformations Electronic Notes in Theoretical Computer Science Volume 152, 27 March 2006, Pages 125-142 Proceedings of the International Workshop on Graph and Model Transformation (GraMoT 2005), 2006.

[14] Saaty, T.L. The Analytical Hierarchical Process. McGraw-Hill, 1980.

[15] Schmidt, D.C. Model-driven engineering. IEEE Computer 39, 2, 25-31, 2006.

[16] Stahl, T., Völter, M. Model-Driven Software Development Technology, Engineering, Management, John Wiley and Sons, Ltd., Chichester, ISBN: 0470025700, 2006.

De flujos de navegación a Spring Web Flow. Un primer acercamiento a las transformaciones verticales en MWACSL*

A. M. Reina Quintero
Departamento de Lenguajes y Sistemas Informáticos
E.T.S. Ingeniería Informática
Universidad de Sevilla
reinaqu@lsi.us.es

J. Torres Valderrama
Departamento de Lenguajes y Sistemas Informáticos
E.T.S. Ingeniería Informática
Universidad de Sevilla
jtorres@lsi.us.es

M. Toro Bonilla
Departamento de Lenguajes y Sistemas Informáticos
E.T.S. Ingeniería Informática
Universidad de Sevilla
mtoro@lsi.us.es

Resumen

MWACSL es una propuesta orientada a aspectos y dirigida por modelos para el desarrollo de aplicaciones web, cuyo acrónimo se obtiene de la combinación de las iniciales de las principales áreas de investigación en las que se enmarca (Model-driven, Web, Aspect-oriented, Concern Specific Languages). La arquitectura de una aplicación MWACSL presenta dos niveles de separación de conceptos. Horizontalmente, los distintos aspectos técnicos que componen la aplicación se separan conceptualmente, y de forma explícita, mediante lenguajes específicos de dominio, de forma que un aspecto técnico puede especificarse mediante uno o varios lenguajes específicos de dominio. Verticalmente, la separación se realiza inspirándose en la arquitectura MDA. Así se consigue una separación de los modelos que son independientes de plataforma de los que son dependientes de la misma. Este artículo se centra en una parte de esta arquitectura, el paso de un modelo independiente de plataforma a uno dependiente de la misma, y en un aspecto técnico concreto, la navegación. El objetivo principal del artículo es doble, por una parte, realizar una primera categorización de los distintos tipos de transformaciones que pueden tener lugar en una arquitectura MWACSL, y, por otra, especificar un caso concreto de uno de estos tipos transformaciones (la generación de flujos de Spring Web Flow a partir de modelos de flujos de navegación).

*Este trabajo se ha financiado parcialmente por el *Ministerio de Ciencia y Tecnología* y fondos FEDER: TIN-2007-67843-C06-03 y TIN2007-64119.

1. Introducción

Algunos de los primeros trabajos publicados relacionados con la orientación a aspectos se centraban en describir una serie de lenguajes específicos de dominio (DSL's) para especificar algunos de los aspectos que componen un sistema. Así, por ejemplo, COOL y RIDL [20], especificaban la sincronización y la coordinación, respectivamente. Sin embargo, la comunidad optó por lenguajes de propósito general, tales como AspectJ [24]. Los principales inconvenientes que existían para tratar con DSL's y que hicieron que la comunidad optara por lenguajes de aspectos de propósito general eran: (1) el importante esfuerzo necesario para implementar un nuevo lenguaje de aspectos para un dominio específico (DSAL); (2) la dificultad para extender un *weaver* realizado ad-hoc para un lenguaje de aspectos de dominio específico; y, (3) la imposibilidad de combinar *weavers* para diferentes DSAL's. Por contra, el trabajar con este tipo de lenguajes tiene importantes beneficios, ya que proporcionan una representación declarativa de los constructores más importantes relacionados con el aspecto que se está tratando, permitiendo así razonar mejor sobre los conceptos y detectar errores a