Estado del arte para las pruebas de software aplicadas en la transformación de modelos

Autores

Johan Alexis Ruiz Echavarría

Flor Katerine Vásquez Hernández

Título otorgado

Especialista en Ingeniería de software

Asesor del trabajo

Jesús Andrés Hincapié Londoño

Especialización en Ingeniería de software

Facultad de ingenierías

Medellín

2014

Estado del arte para las pruebas de software aplicadas en la transformación de modelos

Johan Alexis Ruiz Echavarria[1], Flor Katerine Vásquez Hernandez[2]

[1] Especialización en Ingeniera del Software-Universidad de Medellín, *jare82@gmail.com*
[2]Especialización en Ingeniera del Software-Universidad de Medellín, *katerinevh@gmail.com*

Resumen

A pesar de que el paradigma del desarrollo dirigido por modelos no es un concepto relativamente nuevo y que la implementación de la arquitectura dirigida por modelos no ha sido masificada ampliamente, en contraste la comunidad relacionada con la industria del desarrollo del software ha mostrado un creciente interés por estos nuevos paradigmas, acreditando el aporte que este modelo brinda en el mejoramiento de la calidad del software y la agilidad en la producción y mantenimiento de éste. Sin embargo, se ha adelantado muy poco con respecto al estudio de las técnicas de verificación y validación de las transformaciones bidireccionales que deben darse entre los modelos y código fuente, mediante la aplicación de los conceptos de MDA, MDD, los lenguajes restricciones (OCL) y lenguajes de transformación de Modelos. Dado lo anterior, este artículo tiene como objetivo realizar una revisión al estado actual en el desarrollo de técnicas de aseguramiento de la calidad de las transformaciones entre modelo y código, es decir, los métodos propuestos para la revisión de la correcta traducción de los modelos a través de los lenguajes de transformación, y el código fuente obtenidos.

Palabras clave: V&V en transformación de modelos, Transformación de modelos, Testing en transformación de modelos, Lenguajes de transformación de modelos, métodos formales.

1 Introducción

En el contexto del desarrollo de software enfocado en la arquitectura dirigida por modelos (MDA) se busca una implementación clara y concisa de los requisitos del cliente que cumplan con las características y expectativas del mismo. Esto se logra teniendo en

cuenta la optimización del proceso de desarrollo de software, evitando el desgaste que producen las actividades de diseño, codificación y las pruebas exhaustivas requeridas.
La diferencia notable en esta forma de desarrollo es la generación automática de entregables a partir de herramientas que permiten generar y transformar modelos, lo cual implica la necesidad imperativa de contar con un alto nivel de conocimiento acerca del dominio del negocio, con el fin de lograr que las propuestas de solución a los problemas planteados sean acordes a este dominio.

De acuerdo a lo anterior, el proceso central de MDA está centrado en la transformación de modelos que parten de un modelo inicial y este es transformado de forma iterativa a través de una arquitectura de capas que refina su resultado.
Pero la transformación de modelos no está exenta de errores y fallas debido a la complejidad de la transformación de los modelos y a la posibilidad de que las reglas de transformación no se encuentren bien definidas conceptualmente. Además, no se puede asegurar la carencia de errores en un modelo de dominio del negocio creado mediante la utilización de una herramienta de modelado específico de dominio y un lenguaje de transformación, ya que esta suposición iría en contra de uno de los principios de las pruebas de software: “El proceso de pruebas no puede demostrar la ausencia de defectos”. ISTQB (2011).
Si existen errores y estos no son detectados a tiempo, se corre el riesgo de que los modelos generados de las transformaciones se reutilicen conservando estos errores y en una etapa avanzada del proyecto será más difícil y costoso dar solución a los mismos.
Por tal razón, es necesario presentar y verificar las técnicas, metodologías y prácticas que pueden ser aplicadas en la realización de pruebas de modelado y la transformación de modelos, entre las cuales se encuentran:

- Pruebas funcionales sobre el modelo utilizando técnicas de cala blanca y caja negra.
- Pruebas utilizando OCL para el manejo de las restricciones en el modelo, teniendo en cuenta la abstracción para la descripción del problema de dominio, la consistencia del modelo y el soporte de la generación del código de acuerdo al modelo.
- Pruebas con métodos formales como model cheking para verificar y crear grafos dirigidos.

Gracias a la revisión de los anteriores conceptos se logrará hacer un acercamiento al proceso de la transformación y agregar un punto de partida para analizar las técnicas de pruebas de software mencionadas.

Al evaluar cada uno de estos ítems se debe tener en cuenta el ciclo de vida de un desarrollo basado en arquitectura orientada por modelos (MDA), y las actividades realizadas en la transformación de datos, ya que estos elementos son el eje central de la investigación para establecer marco de referencia que permitan verificar que el proceso de pruebas se realice de forma satisfactoria y el modelo cumpla con las expectativas funcionales.

En este artículo se presenta de forma secuencial el proceso de transformación y pruebas de software realizadas al mismo, estructurando la información de la siguiente manera: Se presentará en el contexto del ciclo de vida de la arquitectura dirigida por modelos el punto en el cual interviene la transformación del modelo. Luego se mostrará al lector revisión de los temas correspondientes a verificación y validación de software representado en el testing de software y su aplicación en la transformación de modelos, empezando por las técnicas de caja blanca, aplicadas a los lenguajes de transformación y luego las restricciones de los lenguajes para las transformaciones, complementando con las técnicas formales de Model Checking. Por último se presentarán las conclusiones del trabajo realizado sobre la revisión de la literatura sobre las pruebas para la transformación de modelos.

2 Método

La metodología investigativa se fundamentará en la documentación a través de la consulta de material bibliográfico como artículos, libros y trabajos de investigación existente sobre el tema de interés.

A su vez, se hará revisión de algunas herramientas disponibles y un acercamiento a la generación de desplegables, a través de su transformación y la aplicación de las técnicas y metodologías obtenidas en la investigación para la realización de pruebas de calidad.

2.1 Preguntas de investigación

¿Cuáles son los métodos propuestos para realizar pruebas funcionales en la arquitectura de modelos, que intervienen en la transformación de los mismos?

¿Cuáles son los puntos más críticos en la transformación de modelos que requieren la verificación y validación en un ciclo de vida de software?

¿Cómo se deben trabajar las técnicas tradicionales de caja blanca y caja negra, para adaptarse a la necesidad de generar casos de prueba para que permita verificar la construcción y transformación de modelos y metamodelos?

2.2 Criterios de inclusión y exclusión

En las pruebas de software se tendrá en cuenta la información de artículos y trabajos que conciernen a las pruebas funcionales, mediante pruebas de caja blanca se requiere evaluar los lenguajes de transformación como describen las transformaciones.

Para el caso de la transformación de modelos por métodos formales, se requiere conocer cuáles son los métodos de verificación por los cuales se orienta a dar solución al problema de calidad en las transformaciones.

A continuación los criterios necesarios para obtener la información del caso de estudio:

Criterios de búsqueda	
Palabras clave	▪ *Transformación de modelos* ▪ *Lenguaje de transformación de modelos* ▪ *Lenguajes de restricciones (OCL)* ▪ *Pruebas de software –Testing* ▪ *Pruebas funcionales* ▪ *Métodos formales AND Model Cheking*
Cadenas básicas de búsqueda	▪ *Model Driven Architecture AND Model Transformation* ▪ *Model Transformation AND (CIM OR PSM)* ▪ *Model Transformation AND (Software testing OR (Black Box AND white Box))* ▪ *Model Driven Architecture AND (Model Transformation AND Software testing)*
Selección en documento.	*Resumen, abstract y palabras clave*
Tipo de fuente de información	▪ *Artículos y revistas especializadas en ingeniería de software.* ▪ *Material bibliográfico sobre DDM,Model Checking y recursos con información certificada disponibles en Internet.*

2.3 Trabajos seleccionados

En esta sección, en la tabla 1 se presenta el material de apoyo para la construcción del artículo.

Tabla 1. Trabajos seleccionados

ID	Título	Autores	Fecha	Fuente
1	Validation of Model Transformations - First Experiences using a White Box Approach	Jochen M. Kuster Mohamed Abd-El-Razik	2006	http://modeva.itee.uq.edu.au/accepted_papers/main.pdf#page=67 Visitada: 08/10/2013
2	OCL contracts for the verification of model transformations.	Eric Cariou, Nicolas Belloir, Franck Barbier, and Nidal Djemam	2009	http://web.univ-pau.fr/~ecariou/papers/ocl-models09.pdf Visitada 29/11/2013
3	A formalism for describing modeling transformations for verification	M. Asztalos, L. Lengyel, and T. Levendovszky.	2009	http://www.model-based-testing.de/modevva09/ Visitada: 29/11/2013
4	*A Lightweight Approach for the Semantic Validation* of Model Refinements.	C. Pons D. Garcia	2008	http://www.lifia.info.unlp.edu.ar/papers/2008/Pons2008.pdf. Visitada: 17/11/2013
5	The Foundation Level Syllabus	International Software Testing Qualifications Board (ISTQB)	2011	http://www.istqb.org/downloads/finish/16/15.html Visitada: 17/11/2013.
6	OMG Object Constraint Language (OCL) v 2.3.1	OMG.ORG	2012	http://www.omg.org/spec/OCL/2.3.1/PDF/ Visitada 17/11/2013

3 Evaluación de los trabajos seleccionados

3.1 Criterios de evaluación

Con el fin de evaluar cada uno de los trabajos y documentos que se convierten en el insumo para la producción de este artículo, se presentan los siguientes criterios de evaluación, los cuales se aplican teniendo en cuenta el eje fundamental del problema.

***Criterio1 - Casos de prueba**:* El artículo establece técnicas para la construcción de casos de prueba a ser implementados en la validación de las transformaciones.

Criterio2 - Técnicas para la realización de pruebas de caja negra: Definición y aplicación de métodos de certificación mediante la realización de pruebas de caja negra.

Criterio 3 - Pruebas de caja blanca: Aplicación de las técnicas de caja blanca para evaluar la reglas de transformación y las plantillas generadas a partir de la misma.

Criterio4 – Lenguaje de restricciones: El documento hace referencia al uso de OCL para definir precondiciones y restricciones sobre los objetos con el fin de realizar las respectivas pruebas de las transformaciones.

Criterio 5 – Transformación de Modelos: La fuente hace referencia sobre el manejo de los modelos y metamodelos en una arquitectura específica.

Criterio 6 – Métodos Formales: La fuente hace referencia sobre los métodos formales y de Model Checking propuestos para simulaciones.

3.2 Resultado de la evaluación

De acuerdo a los criterios mencionados anteriormente, se presentan los resultados de la evaluación teniendo en cuenta la siguiente escala de calificación por criterio:

3.3 Escala de calificación por criterio

Calificación y porcentaje de cumplimiento	**Descripción**
***5:** 90 -100 %*	Los temas tratados en el trabajo cumplen en su totalidad con el criterio especificado, detallando métodos, técnicas o tecnologías que hacen referencia al problema de investigación.
***4:** 70-89%*	Los temas tratados en el trabajo cumplen en gran parte con el criterio especificado. El manejo del tema no es completo en su definición pero aporta elementos importantes detallando métodos,

	técnicas o tecnologías que hacen referencia al problema de investigación.
3: *50-69%*	Los temas tratados en el trabajo cumplen de forma parcial con el criterio especificado debido a que los temas no se manejan en profundidad, por tanto solo permiten en base a estos orientar la búsqueda para profundizar en los temas concernientes al problema.
2: *20-49%*	Los temas tratados en el trabajo mencionan de forma breve el criterio especificado sin hacer énfasis en el tema, detallando aspectos de utilidad para el manejo del problema de investigación.
1: *1-19%.*	Los temas tratados en el trabajo no cumplen con el criterio especificado.

3.4 Características más relevantes de cada trabajo

Tabla 2. **Resumen de la evaluación**

		Criterios de evaluación					
		Criterio 1	Criterio 2	Criterio 3	Criterio 4	Criterio 5	Criterio 6
Trabajos	ID1	4	4	4	3	3	1
	ID2	2	3	2	5	4	2
	ID3	3	2	3	3	4	5
	ID4	4	3	4	4	4	5
	ID5	5	5	5	1	1	1
	ID6	1	1	1	5	2	2

ID1- Validation of Model Transformations - First Experiences using a White Box Approach: Este artículo recopila las experiencias o lesiones aprendidas por los autores luego de la realización de varios procesos de validación de modelos de transformación. A su vez, los autores proponen varias técnicas que pueden ser utilizadas en la construcción de casos de prueba.

ID2- OCL contracts for the verification of model transformations: Este artículo presenta el uso de OCL para realizar la verificación de modelos, basado en las especificaciones del modelo y en la independencia de la plataforma para realizar dicha verificación. Para ello hace referencia a la creación de contratos de la transformación, escritos en OCL.

ID3- A formalism for describing modeling transformations for verification: Este documento está centrado en la descripción del uso de métodos deductivos que permiten mediante formulas y el análisis de propiedades, llevar a fin termino la transformación de un modelos, teniendo en cuenta las precondiciones y condiciones posteriores de los mismos.

ID4- A Lightweight Approach for the Semantic Validation of Model Refinements: Este artículo presenta técnicas utilizadas en métodos formales para realizar la verificación de los modelos, por medio de OCL y Model Checking que indican al detalle el funcionamiento de cada técnica. También realiza un breve comparativo para el testing tradicional.

ID5 - The Foundation Level Syllabus: Este documento perteneciente al ISTQB, es una guía sobre las técnicas de pruebas de software estándar a nivel internacional. Incluye el manejo de pruebas funcionales, diseño de casos de uso y cobertura de pruebas para cualquier tipo de desarrollo de software.

ID5 - OMG : Este documento perteneciente al OMG describe los principales términos y definiciones asociados con OCL, el uso del lenguaje de Restricciones , su sintaxis y el uso del mismo tanto en UML y en metamodelos definidos.

4 Discusión

Para la verificación y validación de modelos es necesario definir como punto de partida la transformación del modelo, el cual está conectado con la pregunta:
”¿Cuáles son los métodos propuestos para realizar pruebas funcionales en la arquitectura de modelos, que intervienen en la transformación de los mismos?”.
Es aquí donde las metodologías de V&V tales como: Testing con técnicas de caja negra y caja blanca y al otro lado métodos formales como Model Checking se incorporan en la transformación del modelo, empezando por la especificación de las restricciones con OCL para construir la definición de reglas de transformación.

Los autores Pons & García (2008) proponen en cada una de sus metodologías realizar la verificación dinámica mediante el uso de OCL, en este punto se observan diferencias en cada metodología, ya que Model Checking descrito por Pons & García (2008) se concentra en establecer un conjunto de estados para realizar una simulación, teniendo presente que las operaciones y restricciones serán definidas con OCL. Para la representación de los estados varios autores confluyen en el uso de un grafo dirigido.

Dado lo anterior, una ventaja en el uso de OCL para realizar la verificación dinámica radica en la expresividad del lenguaje, ya que permite definir precondiciones y condiciones posteriores. A partir de esta especificación se generan las operaciones automáticamente y por ende las condiciones y restricciones para la simulación.

En comparación con el Model Checking, las técnicas de testing adoptan el lenguaje de restricciones acorde a las necesidades del modelo inicial, así como lo expresa los autores Kuster & Abd El Razik (2006) el cual enfoca el manejo de OCL según OMG.ORG (2011) para la construcción de casos de prueba de caja blanca, ya que el manejo de restricciones es un factor clave para la transformación de modelos, en especial de metamodelos que son más susceptibles de presentar errores por restricciones de integridad mientras ocurre la iteración de varias reglas de transformación. Observamos que este enfoque particularmente va muy ligado a la forma de realizar las pruebas tradicionales, de acuerdo Kuster & Abd-El-Razik (2006) , la generación de casos de prueba para la transformación de un modelo, debe realizarse acorde a los criterios de cobertura, para el caso de pruebas de caja blanca se traduce en el manejo de reglas de

transformación y plantillas para los meta- modelos que permiten crear la suite de casos de prueba.

Para este caso se considera que la adaptación de las pruebas de caja blanca deben ir acorde con las adaptaciones que requiere el caso para MDD, en el cual al tener varias transformaciones, es necesario tratar de garantizar que las restricciones realizadas para un mismo modelo se puedan realizar en diferente orden. También se observa en este punto que uno de las ventajas asociada a los casos de prueba creados con OCL radica en el uso modelos generados con una sintaxis correcta que pueden ser usados como los casos de prueba en todo el proceso de transformación.

En cada uno de estos argumentos se debe encontrar un equilibro que permita definir según el modelo, cuál sería la metodología mas optima para realizar la verificación ya que ninguno de los dos autores parece indicar dentro del ciclo de vida del modelo y aun siendo más específicos, en el ciclo de pruebas cuanto es el tiempo y la complejidad que demanda entender el dominio del problema. Por ende el uso de modelos semánticos con formulas y estados podría demandar mucho tiempo para sintetizar el dominio del problema tanto al analista de pruebas como al personal involucrado en el desarrollo si es el caso, además el uso de casos de prueba de caja blanca es un método más sencillo de implementar pero no puede garantizar una cobertura completa de la transformación, representada como un grafo dirigido , tal como indica ISTQB (2011)

Se observa también la desventaja con Model Checking ya que solo podría usarse en grafos dirigidos no muy grandes, para mantener el control de las operaciones y estados sobre los nodos. Como complemento a las ideas expuestas por Pons & García (2008), Cariou,Belloir, Barbier & Djemam(2009) muestra una ventaja en el uso de OCL para la verificación dinámica de modelos y consiste en la independencia de la plataforma, ya que describe el uso de contratos de software los cuales están escritos en OCL y por tanto permiten realizar una prueba dinámica independiente de cualquier herramienta usada para el modelado y transformación de modelos.

En la literatura presentada también queremos resaltar aspectos dentro de la V&V de modelos donde solo interesa la información del modelo de origen y el resultado esperado para el modelo destino o también casos en los cuales no sea necesario realizar la ejecución de la prueba porque no siempre se contaran con los recursos en herramientas para pruebas de modelos.

Dentro de Model Checking encontramos la verificación de modelos por medio de métodos deductivos, en el cual como mencionábamos anteriormente no requiere de la ejecución de la prueba por cada transformación del modelo ya que según los autores Asztalos, Lengyel & Levendovszky (2009), busca mediante la especificación formal de la transformación derivar los casos de prueba para los metamodelos de origen y destino y por ende esta verificación se realiza de forma manual. Esto es muy similar al trabajo que se realiza con las pruebas de caja negra, de acuerdo al ISTQB (2011). A la luz de esta información podemos deducir posibles dificultades en cuanto al tiempo de análisis para extraer los casos de prueba y además en su aplicación, dado que no es posible generar una cantidad considerable de casos de prueba para dar confiabilidad a una aplicación critica donde se depende de que la sintaxis del modelo origen y del modelo destino sean equivalentes. Por ejemplo aplicaciones criticas como las que se presentan en el ensayo del autor (Fernandez et al. 2011).
Observamos al otro lado, en las técnicas de testing con caja negra que existen ciertas similitudes en cuanto al manejo de los casos de prueba ya que también busca modelos de origen validos que cumplan con las restricciones para el modelo con OCL. Para ello y muy ligado del testing tradicional se proponen particiones de equivalencia al metamodelo donde se toman los dominios de los atributos y las cardinalidades de la asociación del metamodelo origen.

Encontramos en este punto que tanto un método deductivo como el uso de pruebas de caja negra conciben la idea de tratar de controlar desde el principio la manera en la cual se realizará la transformación. Aunque no profundizan mucho en la idea de implementar plantillas con restricciones para metamodelos, tal como lo manejan las pruebas de caja blanca, se preocupan por el manejo de varios subdominios para el caso de caja negra y en formulas escritas en lógica temporal para el caso de Model Checking y su método deductivo.

Una desventaja encontrada en el método deductivo propuesto por los autores Asztalos, Lengyel & Levendovszky (2009), es la carencia de personal calificado con experiencia para aplicar las técnicas de especificación formal, aun cuando este tipo de métodos cuenta con cierta madurez en el campo de ingeniería de software. En comparación con el método deductivo, una de las principales ventajas analizadas para las pruebas de caja negra consiste en que es más fácil identificar los posibles valores para las particiones de

equivalencia, debido que en el metamodelo de origen es fácil de identificar las clases y propiedades que permiten dar un cubrimiento mínimo a un fragmento del modelo.

5 Conclusiones

Es este trabajo se ha realizado un recorrido por cada una de las metodologías para la verificación y validación en la transformación de modelos, teniendo en cuenta diferentes propuestas de autores que exponen sus ideas para dar soporte al planteamiento inicial propuesto sobre la forma en que se deben abordar este tipo de pruebas.

Se ha abordado el tema de las metodologías de pruebas de testing y de Model Checking, mostrando en cada una de ellas sus ventajas, desventajas y un breve análisis de su implementación. Para el caso de OCL y la implementación de operaciones y restricciones de modelos se realiza el comparativo entre las pruebas de caja blanca y OCL en Model Checking, cumpliendo con el objetivo de representar el interés de cada una de estas metodologías.

También se abordaron las pruebas centradas en la información del modelo de origen y el resultado esperado para el modelo destino. Por otro lado los casos en que no sea necesario realizar la ejecución de la prueba.

Para estos casos se presentan las pruebas de caja negra y el método deductivo respectivamente, que permiten realizar la verificación y validación del modelo sin enfocarse en la forma como se realiza internamente la transformación.

Se puede concluir que al tratar el tema de verificación y validación de modelos, es un tema obligatorio definir cuáles son las estrategias a llevar a cabo para lograr una ejecución de pruebas del modelo y además de permitir la mayor cobertura posible.

Las técnicas tradicionales de pruebas como las de caja negra y caja blanca, se han adaptado a las necesidades de las transformaciones sucesivas, obteniendo de las reglas de transformación y OCL los elementos necesarios para la creación de casos de prueba.

Estas metodologías solo brindan la seguridad sobre los elementos analizados para la ejecución de los casos de prueba, dejando otros factores que pueden incidir en la estructura y resultado esperado de la transformación. Por otra parte estas metodologías permiten al tester o desarrollador crear casos de prueba con un alto conocimiento del

dominio del problema ,dando como resultado casos de prueba más efectivos en su diseño y ejecución.

Para el caso de las pruebas basadas en Model cheking se presentan conceptos diferenciadores de las otras pruebas como el manejo de estados y lógica temporal que permiten mayor precisión en las operaciones y resultados esperados en la transformación.

A su vez el método deductivo de Model cheking representa una mayor dificultad en su aprendizaje y experiencia, por tanto para un proyecto que no requiera la complejidad para el uso de métodos deductivos u OCL, se recomienda los métodos tradicionales de pruebas los cuales ya tienen la madurez y la aplicación inmediata sobre los modelos, artefactos y productos.

6 Referencias

C.A Fernandez, M. Ambrosio,G.Andrade,G.Cruz,M.J.Román,H.Sánchez.(2011). Métodos formales aplicados en la industria del software. Temas de Ciencia y Tecnología *. 15 (13), pp.3-12*

C. Pons ,D.Garcia, (e.g. 2008). A Lightweight Approach for the Semantic Validation of Model Refinements . *CONICET (Consejo Nacional de Investigaciones Científicas y Técnicas) Buenos Aires, Argentina*. (), pp.2-9

E. Cariou, N. Belloir, F. Barbier, N. Djemam, (2009). OCL contracts for the verification of model transformations. *Universit´e de Pau et des Pays de l'Adour / Liuppa,Pau Cedex France*. (e.g. 2), pp.5-8

ISTQB.ORG (e.g. 2011). *Foundation Level syllabus*. [ONLINE] Recuperado de: http://www.istqb.org/downloads/finish/16/15.html. [último acceso 29 Noviembre 2013].

J. M. Kuster, M Abd-El-Razik. (2006). Validation of Model Transformations – First Experiences using a White Box Approach. Models in software Engineering, 9-12.

M. Asztalos, L. Lengyel, and T. Levendovszky. , (2009). A formalism for describing modeling transformations for verification.. *In Proceedings of the 6th International Workshop on Model-Driven Engineering, Verification and Validation, ACM*. (), pp.

OMG.ORG. (2012). *OMG Object Constraint Language (OCL) v 2.3.1*. [ONLINE] Recuperado de:: http://www.omg.org/spec/OCL/2.3.1/PDF/. [último acceso 29 Noviembre 2014].