Una aproximación a las pruebas de aplicaciones Web basadas en un contexto MDWE

Arturo H. Torres, María J. Escalona, Manuel Mejías, Javier J. Gutiérrez

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla,
Avd. Reina Mercedes sn. 41040 Sevilla, España
arturoh.torres.exts@juntadeandalucia.es
{escalona,risoto,javierj}@lsi.us.es

Resumen. La garantía de la calidad (***Q**uality **A**ssurance*) es el proceso de verificar si los productos o servicios satisfacen las expectativas del cliente. Las pruebas forman parte de este proceso. No obstante, los investigadores y profesionales todavía están tratando de encontrar formas efectivas para probar las aplicaciones Web. Una de ellas está relacionada con el paradigma MDA (***M**odel-**D**riven **A**rchitecture*). Este trabajo presenta un estudio comparativo de las propuestas existentes con este en-foque, y a partir de éste, propone una primera aproximación de una técnica de pruebas basada en MDWE (***M**odel-**D**riven **W**eb **E**ngineering*).

Palabras clave: MDA, MDWE, Pruebas basadas en modelos, Modelos nave-gacionales.

1 Introducción

Las pruebas de software son un término bastante amplio y abarcan una extensa gama de actividades muy variadas, desde las pruebas realizadas por el desarrollador, de una pequeña pieza de código (pruebas unitarias), hasta la validación del cliente de un gran sistema de información (pruebas de aceptación). En todas estas fases, los casos de prueba pueden ser concebidos con objetivos muy variados, tales como validar si existen desviaciones en los requisitos del usuario, evaluar la conformidad de una especificación, o de entradas maliciosas, medir atributos como el desempeño o usabilidad, etc.

Para aclarar y organizar todos estos términos y con el objetivo de mostrarlos en una vista unificada, Bertolino [3] presenta una clasificación de los problemas comunes y de los muchos significados de las pruebas de software. En este artículo, abordamos tres desafíos de los enunciados por Bertolino, al considerarlos adecuados para nuestra propuesta de investigación. Los desafíos abordados son el obtener oráculos de pruebas eficientes, conseguir pruebas 100% automáticas y efectuar pruebas basadas en modelos. Y específicamente dentro de las pruebas basadas en modelos, tratamos el modelado de las pruebas con el paradigma MDA, ya que los niveles de abstracción de MDA pueden también aplicarse al modelado de pruebas [10]. Además, debido a que el interés de este trabajo son las aplicaciones Web, estos desafíos estarán orientados hacia este tipo de software. Es decir, tomar el desafío de obtener pruebas 100% automáticas, con los adecuados oráculos de pruebas, en un contexto de meta-modelos.

Además, es en el ámbito de la Ingeniería Web donde se ha evaluado la necesidad de estudiar de manera concreta una característica del software, que, en los últimos años, está definiéndose como crítico dentro del proceso de desarrollo: la navegación [5]. Este trabajo propone una aproximación de pruebas basada en los modelos navegacionales. La estructura de este artículo continúa con la sección 2, presentando los trabajos relacionados. Seguidamente en la sección 3 se presenta un breve análisis de las propuestas existentes con el objetivo de identificar las oportunidades de investigación. En seguida, a partir de las oportunidades, la sección 4 presenta una aproximación de un método de

pruebas de software basado en el paradigma MDWE. Finalmente la sección 5 presenta las conclusiones y trabajos futuros.

2 Trabajos Relacionados

Esta sección presenta los trabajos relacionados con los modelos de pruebas de software que utilizan el enfoque MDA. Entre las propuestas de empresa que presentan las pruebas basadas en modelos, sólo algunas tienen un enfoque MDA. Estas propuestas están especificadas en la Tabla 1, y están marcadas en la última columna de la tabla: Objecteering Software [15], Tau Generation 2 [24] y Test Designer [25].

Objecteering Software [15] combina modelado UML, producción de código, *debugging* y pruebas de aplicaciones Java en un ambiente simple. Es una herramienta orientada a modelos, soportando la tecnología MDA. La herramienta Test Designer v3.3 [25] automatiza el diseño de pruebas, incluyendo diversas fases. Test Designer genera todos los casos de prueba a partir de la especificación de un modelo funcional, por ejemplo, UML.

Tabla 2. Tabla comparativa de herramientas orientadas a modelos.

Herramienta	Validación del modelo	Métricas	Anti patrones	Navegación	Visualización del modelo	Pruebas con MDA
All Fusión Component Modeler [1]	√	x	x	x	x	x
ArcStyler [2]	√	x	x	x	x	x
iUML [11]	√	x	x	x	√	x
NetBeans [14]	N/A	N/A	N/A	N/A	N/A	x
Objecteering Software [15]	√	√	x	x	x	√
Poseidon [18]	√	x	x	√	√	x
Rhapsody [21]	√	x	x	√	x	x
SD Metrics [23]	x	√	x	x	√	x
Tau Generation 2 [24]	√	x	x	√	√	√
Together [27]	√	√	√	√	x	x
WayPointer [31]	√	x	x	N/A	N/A	N/A
XDE [20]	√	x	x	x	x	x
Test Designer [25]	√	x	x	x	x	√

La herramienta Test Designer v3.3 implementa el concepto de Smart Testing. Smart Testing son pruebas que, basándose en la teoría o en la experiencia, tienen una alta probabilidad de detectar clases específicas de errores; son pruebas dirigidas a tipos específicos de errores. Además, Test Designer v3.3 soporta las pruebas basadas en modelos.

Finalmente, Telelogic TAU Generation2 [24] representa la generación avanzada de desarrollo y herramientas de pruebas, soportando los estándares de la industria para sistemas visuales y desarrollo de software (U2TP) e integración de pruebas (TTCN-3). El equipo de Telelogic proporciona un método que automatiza las actividades de pruebas cubriendo la especificación, desarrollo y ejecución de las pruebas. U2TP es seleccionado como lenguaje de modelado para la especificación de casos de prueba. Los modelos son entonces transformados al lenguaje TTCN-3, el cual es usado para describir los casos de prueba ejecutables. Por otro lado, tenemos las propuestas académicas relacionadas con las pruebas de software basadas en modelos y en un contexto MDA. Existen varias propuestas, por ejemplo [4] [6] [17], que utilizan este enfoque; sin embargo, para los propósitos de este apartado se ha escogido tres propuestas [7] [32] y [16] que representan los conceptos comunes existentes. Es decir, aquellas propuestas que son más representativas. Dichas propuestas utilizan los siguientes conceptos: la obtención de modelos de pruebas a través de las transformaciones U2TP y TTCN-3. Dai [7] introduce una metodología acerca de cómo usar el profile U2TP a fin de transformar un modelo de diseño de sistema UML en modelos de pruebas. Para la formalización de la propuesta metodológica, son consideradas las reglas de transformación Query/View/Transformation (QVT) [19]. Zander et al.

[32] presentan un método para derivar automáticamente las pruebas ejecutables a partir de diagramas UML, usando el *profile* U2TP. Se presenta una transformación entre las especificaciones de U2TP [29] usadas para representar PITs y TTCN-3 [28]. Las transformaciones son especificadas como reglas de transformación entre el meta-modelo U2TP [29] y el meta-modelo TTCN-3 [28]. Posteriormente, la salida generada es completada y compilada en código de pruebas ejecutables en Java [26]. Los meta-modelos U2TP y TTCN-3 son definidos por los modelos Meta Object Facility (MOF) [12]. Las reglas de transformación proporcionadas en este trabajo, definen relaciones entre las meta-clases origen y destino de estos meta-modelos.

Pérez et al. [16] presenta una propuesta para pruebas en el contexto de la ingeniería dirigida por modelos. A partir de los modelos de diseño del sistema en UML, se propone realizar transformaciones a modelos de prueba basados en el perfil de pruebas de UML. Para que la generación de los casos de prueba sea automática, se define una extensión del meta-modelo de UML, de forma que se puedan anotar los diagramas de secuencia con información que, luego, pueda ser utilizada para generar el oráculo de pruebas. Esta información es anotada en OCL como pre y post-condiciones en el diagrama. Se presenta una propuesta para la generación automática de casos de prueba en el contexto de MDA, basada en el meta-modelo de UML y su perfil de pruebas, realizando transformaciones desde los modelos UML al modelo de pruebas, utilizando como modelo de descripción de comportamiento del sistema el diagrama de secuencia de UML. Dentro de la propuesta se aborda la generación automática de los oráculos de las pruebas ya que éstos son dependientes del dominio de la aplicación.

3 Discusión

El objetivo de este apartado es identificar las oportunidades de investigación a partir de los trabajos relacionados expuestos. Esta identificación se realiza teniendo en cuenta las ventajas y desventajas de cada propuesta. En primer lugar, la propuesta de Dai introduce una metodología acerca de cómo usar el profile U2TP a fin de transformar un modelo de diseño de sistema UML en modelos de pruebas. Presenta la definición de las reglas de transformación, pero el inconveniente es que aún no están totalmente completadas. Por lo tanto, es un frente abierto para futuros trabajos. También debido a la falta de herramientas de soporte para UML 2.2 y U2TP, no están en condiciones para probar las reglas de transformación. Dai expresa que en sus trabajos futuros, investigarán en herramientas que soporten los conceptos U2TP y la derivación automática de modelos de diseño de prueba a partir de modelos de diseño. Por tanto, no se contempla el desarrollo de una herramienta de automatización del proceso.

Por otro lado, la propuesta de Zander et al. [32] presenta un método para derivar automáticamente las pruebas ejecutables a partir de diagramas UML, usando el profile U2TP. En la propuesta se presenta una transformación entre las especificaciones de U2TP [29] usadas para representar PITs y TTCN-3 [28]. La principal ventaja de esta propuesta es que proporciona las reglas de transformación entre el meta-modelo U2TP y el meta-modelo TTCN-3.

Otra de las ventajas de este método es que presenta un ambiente de ejecución automático, Eclipse fue usado para demostrar la viabilidad de este trabajo junto al plug-in UML 2.0 [8] y otro desarrollado para soportar U2TP. También usan un plug-in para soportar TTCN-3. Los modelos con conceptos U2TP la integran con la plataforma Eclipse, como lo hace el equipo de Objecteering. También se desarrolla transformación de modelos de U2TP para TTCN-3 como los ofrecidos por Telelogic, pero en este trabajo se define las reglas en el nivel de meta-modelo usando los métodos disponibles en Eclipse para implementarlos.

Finalmente, la propuesta de Pérez et al. [16], presentan una propuesta para pruebas en el contexto de la ingeniería dirigida por modelos. A partir de los modelos de diseño del sistema en UML, se propone realizar transformaciones a modelos de prueba basados en el perfil de pruebas de UML. Esta propuesta tiene como ventaja principal el tratamiento de los oráculos de prueba, del cual se puedan derivar los datos de prueba en forma automática. Para esto han definido una extensión del meta-modelo de UML donde se expresan las pre y post-condiciones de cada diagrama de secuencia

mediante el lenguaje OCL. Esta propuesta no cuenta con una herramienta de automatización que valide la propuesta.

A continuación, en la Tabla 2 se presenta todas las propuestas estudiadas, tanto las propuestas empresariales, así como también las propuestas académicas. La tabla muestra las características necesarias que cubren los tres desafíos de pruebas de software que se va a abordar en este trabajo. Con el símbolo √ se señala las propuestas que sí cumplen con la característica especificada. Por el contrario, el símbolo x señala que no las cumplen. El símbolo ψ indica que aunque la propuesta cumple la característica, no la cubre completamente; es decir, está incompleta. El símbolo δ indica que la automatización es semi-automática. Finalmente, el símbolo θ señala que aunque la característica es abordada, son desconocidos los detalles que cubren la característica.

Tabla 2. Cuadro de oportunidades.

		Propuestas académicas			Propuestas empresariales		
		Dai [7]	Zander *et al.* [32]	Pérez *et al.* [16]	Objecteering Software [15]	Test Designer [25]	Telelogic TAU [24]
Desafío 1	Oráculos de prueba	**x**	**x**	√	**x**	**x**	**x**
Desafío 2	Automatización del proceso	**x**	√	**x**	$\surd_{\delta}$	$\surd_{\delta}$	$\surd_{\delta}$
Desafío 3	Enfoque MDA	√	√	√	√	√	√
	Reglas de transformación	$\surd_{\psi}$	$\surd_{\psi}$	x	$\surd_{\theta}$	$\surd_{\theta}$	$\surd_{\theta}$
	Enfoque MDWE	**x**	**x**	**x**	**x**	**x**	**x**
	Basados en navegación	**x**	**x**	**x**	**x**	**x**	**x**

Leyenda: ψ Incompletas δ Semi-automáticas θ Desconocidos

De esta manera, con la ayuda de este cuadro comparativo se identifica las oportunidades de investigación, y a partir de éstas, en el siguiente apartado se plantea la propuesta de este trabajo. La primera oportunidad es referente al tratamiento de los oráculos de prueba, según Bertolino, éste es el principal obstáculo para conseguir una automatización 100% del proceso de pruebas. Los oráculos de pruebas en un contexto MDA, son abordados solamente por Pérez et al., pero sólo con los artefactos de diagramas de secuencia. Entonces, un importante desafío es el conseguir los mecanismos eficientes para obtener un oráculo de pruebas robusto, ya que aún se carece de métodos que nos lo proporcionen. El cuadro también refleja que todas las propuestas enunciadas están basadas en el enfoque MDA, pero no todas proporcionan las reglas de transformación necesarias para este enfoque. Si bien es cierto, las propuestas empresariales cubren esta característica, se desconoce el detalle y naturaleza de éstas. Existen, algunas propuestas académicas que presentan reglas de transformación, pero están incompletas.

La última oportunidad y la más importante para nuestra investigación es que las propuestas, ya sean académicas o empresariales, no están orientadas al desarrollo de aplicaciones Web. Por lo tanto, no utilizan el enfoque MDWE y no están basados en los modelos navegacionales. La siguiente sección presenta nuestra primera aproximación de investigación, en base a las oportunidades planteadas, tomando los puntos fuertes de los trabajos relacionados e intentando rellenar los vacíos metodológicos de dichas propuestas.

4 Método propuesto

El objetivo de esta investigación es mejorar las técnicas de garantía de calidad en el desarrollo de aplicaciones Web. Para ello, nos enfocamos en el desarrollo de una técnica de pruebas de nivel de sistema basada a partir de modelos navegacionales. La hipótesis es que la generación de modelos de

prueba de sistema a partir de modelos navegacionales mediante el paradigma MDWE puede ser usado para mejorar las pruebas de aplicaciones Web. En la Fig. 1 se presenta de manera esquemática la propuesta. Los modelos de requisitos mostrados en la figura, pertenecen a la metodología NDT (Navigational Development Techniques) [9]. Las siguientes fases forman parte de la propuesta:

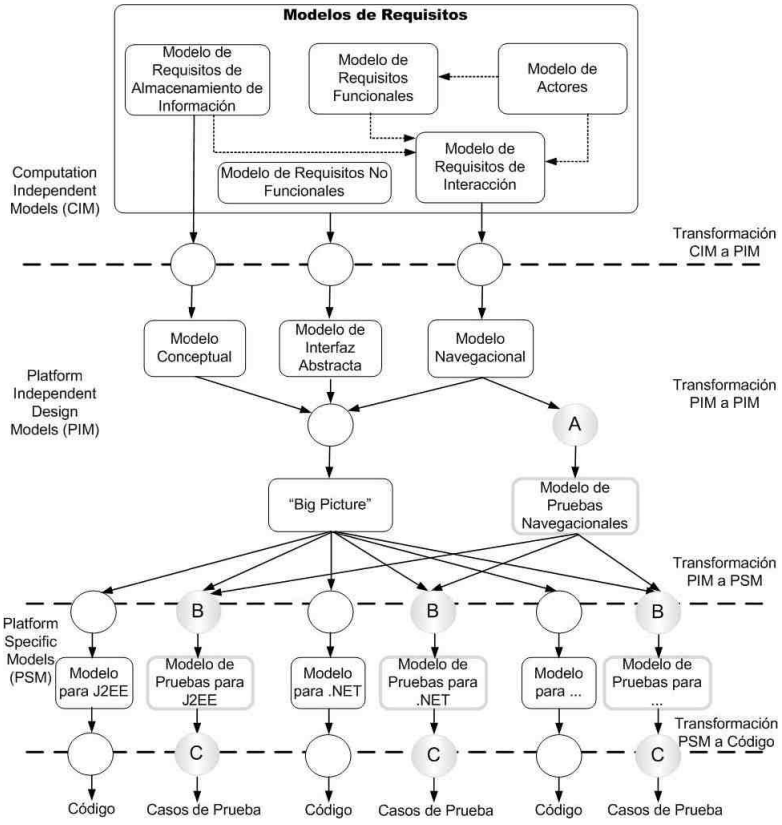

Fig. 1. Propuesta de pruebas en un contexto MDWE

Fase 1: Obtener el modelo de pruebas de sistema basado en transformación PIM to PIM. Se creará los algoritmos necesarios para la obtención del modelo de pruebas de sistema a partir de los modelos navegacionales, esta transformación está enmarcada dentro de los modelos de diseño independientes de la plataforma (PIM), y debido a que tanto el modelo fuente, así como también el modelo destino son PIM, será necesario realizar una transformación PIM to PIM. En la Fig. 1 se muestra la ubicación de esta transformación marcado con la letra A.

Fase 2: Obtener el modelo de pruebas de sistema especifico basado en trans-formación PIM to PSM. En el nivel PIM, el modelo conceptual, el modelo navegacional y el modelo de interfaz abstracta son integrados y transformados en un modelo que representa una visión global del sistema ("Big Picture"). Es así que, una vez obtenido el modelo de pruebas navegacionales del sistema en el nivel PIM, con ayuda de la especificación denotada en "Big Picture", será necesario obtener un modelo de prueba de sistema específico de la plataforma en estudio; por ejemplo, modelos de pruebas para J2EE, .NET, etc. Para conseguir este modelo, se utilizará una transformación PIM to PSM (letra B en la Fig. 1). Se dotará de las reglas y algoritmos necesarios para cumplir el objetivo en cuestión.

Fase 3: Obtener los casos de prueba basado en transformación PSM to Code. Ya obtenido el modelo de pruebas de sistema específico para una plataforma, podremos generar los casos de prueba. Para

ello, se utilizará una transformación PSM to Code (letra C en la Fig. 1), donde también será necesario definir la sintaxis y semántica de los casos de prueba a obtener.

Fase 4: Desarrollar una herramienta de automatización del proceso. Un objetivo de la propuesta es proveer una herramienta que capte los algoritmos realizados y que sea útil para la automatización de todo el proceso. Esta herramienta formará parte de NDT Suite [13]. NDT-Suite es un conjunto de herramientas para aplicar la metodología NDT en entornos prácticos.

5 Conclusiones

Este trabajo ha presentado aspectos importantes relacionados con las pruebas de aplicaciones Web. En la Ingeniería Web se ha evaluado la necesidad de estudiar de manera concreta a la navegación, pues es una característica del software, que, en los últimos años, está definiéndose como crítico dentro del proceso de desarrollo.

También, uno de los aspectos importantes mostrados, son los relacionados con los actuales desafíos en las pruebas de software, y a partir de éstos, se identificaron las oportunidades de investigación relacionadas con el desarrollo del proceso de pruebas en un contexto de transformación de modelos MDWE.

En base a todos los aspectos mencionados, se presentó una propuesta, la cual aprovechando las oportunidades identificadas en el análisis de esta investigación, pretende cubrir los grandes desafíos de las pruebas de software destinados a conseguir un desarrollo seguro y eficiente de las aplicaciones Web. Se evaluará el resultado del proceso propuesto con la metodología NDT [9], la cual surgió como resultado de un trabajo de investigación dentro del grupo.

Actualmente se está evaluando la posibilidad de abordar el desafío de los oráculos de pruebas mediante técnicas heurísticas de la Inteligencia Artificial, entre ellas, las redes neuronales.

Agradecimientos. Este trabajo ha sido apoyado por el proyecto QSimTest (TIN2007-67843-C06 03) y el proyecto RePRIS del Ministerio de Educación y Ciencia (TIN2007-30391-E), España.

Referencias

1. AllFusion Component Modeler, http://www.astrom.se/allfusion
2. ArcStyler, http://www.arcstyler.com
3. Bertolino, A.: Software testing research: Achievements, challenges, dreams. In: FOSE 07: Future of Software Engineering, pp 85-103. IEEE Computer Society, Washington (2007)
4. Busch, M., Chaparadza, R., Dai, Z., Hoffmann, A., Lacmene, L, Ngwangwen, T., Ndem, G., Ogawa, H., Serbanescu, D., Schieferdecker, I., Zander-Nowicka, J.: Model transformers for test generation from system models. Technical report, Fraunhofer FOKUS, Germany and Hitachi Central Research Laboratory Ltd., Japan (2006)
5. Cachero, C., Koch, N.: Conceptual navigation analysis: a device and platform independent navigation specification. In: 2nd International Workshop on Web-oriented Software Technology (IWWOST02), Málaga (2002)
6. Dai, Z.R., Deussen, P.H., Lacmene, L.P., Busch, M., Ngwangwen, T., Herrmann, J., Schmidt, M: Grid.: Automatic test data generation for TTCN-3 using CTE. In: 18th International Conference Software Systems Engineering and their Applications, Paris (2005).
7. Dai, Z.R.: Model-driven testing with UML 2.0. In: Proceedings of the Second European Workshop on Model Driven Architecture, pp. 179–187. University of Kent, UK (2004)
8. Eclipse UML2, http://www.eclipse.org/uml2
9. Escalona M.J.: Modelos y técnicas para la especificación y el análisis de la Navegación en Sistemas Software. PhD thesis, University of Seville, Seville (2004)

10. Gross, H.: Testing and the UML a perfect fit. Technical report, Fraunhofer IESE Report 110.03E, Alemania (2003)
11. iUML, http://www.kc.com/products/iuml.php
12. MOF, http://www.omg.org/technology/documents/formal/mof.htm
13. NDT Suite, http://www.iwt2.org/ndt.php
14. NetBeans, http://www.netbeans.org
15. Objecteering Software, http://www.objecteering.com
16. Pérez, B., Reales, P., García, I., Polo, M.: Propuesta para pruebas dirigidas por modelos usando el perfil de pruebas de UML 2.0. In: Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, Gijón (2008)
17. Pietsch, S., Stanca-Kaposta, B.: Model-based testing with UTP and TTCN-3 and its application to HL7. Technical report, Conquest Potsdam, Germany (2008)
18. Poseidon, http://www.gentleware.com
19. QVT, http://tefkat.sourceforge.net/publications/ad-04-01-06.pdf
20. Rational XDE, http://www.ibm.com/developerworks/rational/products/xde
21. Rhapsody, http://www.telelogic.com/products/rhapsody/index.cfm
22. Schieferdecker, I., Din, G.: A meta-model for TTCN-3. In: Applying Formal Methods: Testing, Performance, and M/E-Commerce. LNCS, vol. 4, pp. 226-245. Springer (2005)
23. SD Metrics, http://www.sdmetrics.com
24. Telelogic Tau, http://www.telelogic.com/products/tau/index.cfm
25. Test Designer v3.3, http://www.smartesting.com/cms/en/explore/products
26. Testing Technologies, http://www.testingtech.de
27. UML Testing Profile, http://utp.omg.org
28. TTCN-3, http://www.ttcn-3.org
29. UML Testing Profile, http://utp.omg.org
30. Unified Modeling Language, http://www.uml.org
31. WayPointer, http://www.jaczone.com/product/overview
32. Zander, J., Dai, Z.R., Schieferdecker, I., Din, G.: From U2TP models to executable tests with TTCN-3 - an approach to model driven testing. In: Testing of Communicating Systems. LNCS, vol. 3502/2005, page. 289–303. Springer, Berlin (2005)