Revista
Española de

Innovación, Calidad e Ingeniería del Software

REICIS

Volumen 4, No. 4, diciembre, 2008

Web de la editorial: www.ati.es
E-mail: reicis@ati.es
ISSN: 1885-4486

Publicado por la Asociación de Técnicos en Informática

Revista Española de Innovación, Calidad e Ingeniería del Software (REICIS)

Contenidos

REICIS

Editorial

REICIS

Este número de diciembre de 2008 de REICIS, en su sección regular de artículos, se nutre de dos contribuciones que son fruto del acuerdo con la organización del Taller sobre Pruebas en Ingeniería del Software PRIS 08 celebrado en Gijón el 7 de octubre de 2008 en el marco de las Jornadas de Ingeniería del Software y Bases de Datos. El proceso incluye la selección de los originales más prometedores para posteriormente solicitar a sus autores una versión mejorada y extendida. Esta versión es finalmente revisada por el comité editorial solicitando las mejoras necesarias para su publicación en REICIS. Para completar la cobertura de esta reunión científica, este número incluye una reseña sobre el taller PRIS'08 a cargo de su responsable, Claudio de la Riva, de la Universidad de Oviedo.

La política de REICIS de potenciación de los acuerdos con eventos relevantes en los que se persiguen objetivos de aplicación práctica de las propuestas a la vez que existe un control apropiado de la calidad científica y técnica de las mismas es esencial para el desarrollo de las disciplinas de ingeniería y calidad del software. Es misión de REICIS facilitar la llegada de la información de estas propuestas a todos los interesados a la vez que resaltar las contribuciones más prometedoras en un medio que garantiza una amplia difusión a la vez que una mayor consideración en las evaluaciones administrativas para las carreras de docentes e investigadores que las simples actas de congresos nacionales.

Luis Fernández Sanz
Juan J. Cuadrado-Gallego
Editores

Presentación

REICIS

Este número de REICIS publica, tras el proceso de revisión de nuestro comité editorial, dos contribuciones extendidas y revisadas seleccionadas de entre las remitidas al taller PRIS'08.

En este caso, el primero de los trabajos publicados corresponde al titulado "TestPAI: Un área de proceso de pruebas integrada con CMMI" cuyos autores son Ana Sanz, Javier Saldaña y Javier García de la Universidad Carlos III de Madrid con al colaboración de Domingo Gaitero de ATOS ORIGIN. En este artículo se presenta el análisis de los distintas propuestas existentes de definición y mejora de procesos (obtenidas de los modelos y estándares internacionales más relevantes) para crear y definir un área de mejora de proceso de pruebas integrada con CMMi en el formato previsto en este modelo.

El segundo trabajo se titula "Propuesta para pruebas dirigidas por modelos usando el perfil de pruebas de UML 2.0" y ha sido elaborado por Beatriz Pérez Lamancha del Centro de Ensayo de Software de Uruguay en colaboración con Pedro Reales Mateo, Ignacio García-Rodriguez de Guzmán y Macario Polo Usaola de la universidad de castilla-La Mancha. Describe una propuesta de pruebas basada en el metamodelo UML para permitir generación de pruebas con un enfoque MDA incluyendo el uso de transformaciones de modelos QVT.

Para completar la información sobre el taller PRIS'07 se incluye una reseña del responsable del mismo, el profesor Claudio de la Riva de la Universidad de Oviedo, que glosa tanto la posición de esta iniciativa como las principales aportaciones de los participantes. La colaboración con iniciativas en esta línea de pruebas de software continuará en próximos eventos a celebrar en España con los que REICIS ya está acordando cauces de colaboración.

Finalmente, en la columna de Actualidad Invitada, es Carlos Fernández Gallo como Jefe de Área de Informática de la Subdirección General para la Economía Digital del Ministerio de Industria, Turismo y Comercio (MITYC) quien ha tenido la gentileza de glosar las principales acciones y líneas de ayuda que el MITYC está impulsando como apoyo a la mejora de la calidad del software en las empresas del sector con especial interés en la promoción de certificaciones.

Luis Fernández Sanz

TestPAI: Un área de proceso de pruebas integrada con CMMI

Ana Sanz, Javier Saldaña, Javier García
Departamento de Informática, Universidad Carlos III de Madrid
{asanz, jsaldana, jgarciag}@inf.uc3m.es

Domingo Gaitero
ATOS - ORIGIN S.A
domingo.gaitero@atosorigin.com

Abstract

Today, there are different organizations which are using CMMI to improve their processes. In many cases, they are interested in improving their testing process in parallel with the improvement of other processes. But CMMI does not provide the necessary support to improve testing process. In this paper, we analyze different testing reference models and propose a new process area to develop testing process improvement. TestPAI is a detailed defined process area which is integrated with CMMI to improve testing process. TestPAI has a structure similar to CMMI and includes all practices related to testing. Moreover, we have implemented TestPAI in a real organization successfully. For this, we defined a new testing process and modified other existing ones. Finally, we evaluated this new process to check its viability.

Key words: Software Testing Process, Testing Process Improvement, Testing Reference Model, CMMI.

Resumen

Actualmente existen numerosas organizaciones que están utilizando CMMI como modelo de referencia para mejorar sus procesos software. Muchas de ellas se encuentran interesadas en mejorar el proceso de pruebas al mismo tiempo que se produce la mejora en el resto de los procesos de la organización. Sin embargo, CMMI no proporciona el soporte adecuado para realizar dicha mejora. En este artículo, se analizan los diferentes modelos de referencia que existen en relación al proceso de pruebas y se propone una nueva área de proceso que proporcione el soporte necesario para llevar a cabo la mejora de dicho proceso. TestPAI es un área de proceso totalmente definida que puede ser integrada con CMMI para mejorar el proceso de pruebas de forma paralela a la mejora de otros procesos software. TestPAI tiene una estructura similar a CMMI e incluye todas las prácticas relacionadas con las pruebas. Además, TestPAI ha sido implementada satisfactoriamente en una organización real, siendo necesario para ello definir procesos nuevos y modificar algunos existentes, y se ha realizado una evaluación de la implantación de TestPAI y un análisis de los resultados.

Palabras clave: Proceso de Pruebas, Mejora del Proceso de Pruebas, Modelo de Referencia de Pruebas, CMMI.

1. Introducción

La calidad de un sistema software vendrá determinada por la calidad del proceso software que lo produce [1]. Análogamente, la calidad de las pruebas software vendrá determinada por la calidad del proceso de pruebas que se utilice. Así, la solución a muchos de los problemas relacionados con las pruebas consiste en tener un proceso de pruebas bien definido, gestionado y controlado, que comenzará con la fase de definición de requisitos y se desarrollará de forma paralela al proceso de desarrollo del software. Un proceso de pruebas bien definido implica:

- Identificar y establecer los objetivos, políticas y estrategia de pruebas que guiarán al proceso.
- Desarrollar y mantener un plan de pruebas que recoja los aspectos necesarios para la gestión y control de las actividades de prueba.
- Definir las especificaciones necesarias para generar los casos de prueba adecuados.
- Identificar y establecer las medidas adecuadas para solucionar los posibles problemas encontrados.

La mejora del proceso de pruebas generará un conjunto de beneficios para la industria del software. Los más destacables son los siguientes [2]:

- Se produce un incremento de la satisfacción del cliente al utilizar un software con una cantidad de errores inferior.
- Se incrementa la eficiencia del proceso de desarrollo.
- Se facilita la definición y cumplimiento de los objetivos de calidad.
- Se incrementa la satisfacción de los trabajadores debido a que se proporcionan herramientas y recursos apropiados para la realización eficiente del trabajo.

Actualmente, existen diferentes empresas que están trabajando con CMMI para mejorar sus procesos. Según el perfil de madurez mundial de las organizaciones intensivas en software [3] elaborado por el Software Engineering Institute (SEI), de aquellas organizaciones (1024 en 2008) que evaluaron sus procesos con respecto al nivel 2 de capacidad establecido por el CMMI para los Procesos de Verificación y Validación, menos del 2% de las organizaciones lograron satisfacer totalmente ese nivel, mientras que las organizaciones que lo lograron satisfacer parcialmente, es decir no en la totalidad de los requisitos marcados por el CMMI, no superan el 4,5%.

Muchas de ellas se encuentran interesadas en la mejora de su proceso de pruebas, sin embargo CMMI no les proporciona el soporte necesario, y la integración de CMMI con otros modelos existentes es muy costosa debido a la incompatibilidad existente entre ellos. Así, el problema al que se enfrentan estas organizaciones es la carencia de un modelo de referencia que se integre totalmente con CMMI para llevar a cabo las actividades de mejora del proceso de pruebas de forma paralela a la mejora de otros procesos de la organización.

La hipótesis de este trabajo es desarrollar un área de proceso de pruebas que contenga todas las practicas relativas a las pruebas y se integre totalmente y de una forma sencilla con CMMI. De este modo, será posible desarrollar de forma eficaz las actividades relativas a la mejora del proceso de pruebas mientras se desarrollan las actividades de mejora relativas a otros procesos implementados dentro de la organización. El area de proceso de pruebas desarrollado permitirá que todas aquellas empresan que utilizan CMMI y desean mejorar el proceso de pruebas puedan alcanzar el objetivo marcado sin necesidad de utilizar un modelo de referencia diferente; es decir mediante una integración sencilla y completa del área de proceso de pruebas con CMMI.

Los objetivos principales de este trabajo son:

1. Definir un proceso de pruebas basado en el área de proceso de pruebas desarrollado.
2. Implantar el proceso de pruebas en una organización que se encuentre trabajando con CMMI y cuyo objetivo sea la mejora del proceso de pruebas.
3. Establecer un método que permita realizar una auto-evaluación del proceso de pruebas implantado y que se encuentre basado en SCAMPI [4].

La estructura de este artículo es la siguiente: en la sección 2 se describe los trabajos existentes en relación a la mejora del proceso de pruebas. La sección 3 especifica y describe el area de proceso que ha sido desarrollado: TestPAI. En la sección 4 se presenta la implementación que se ha realizado de TestPAI en una organización Española. Y, finalmente, en la sección 5 se muetran las conclusiones extraidas tras el desarrollo de este trabajo.

2. Análisis de los Modelos de Referencia de para el Proceso Pruebas

Un modelo de referencia orientado al proceso de pruebas define el marco de referencia necesario para poder determinar las fortalezas y debilidades del proceso de pruebas

implementado en la organización. Es decir, para poder determinar el estado actual del proceso y establecer el plan de mejora, se recogerán las prácticas existentes en la organización y se compararán con las propuestas en el modelo de referencia, por tanto el primer requisito de un modelo de referencia dirigido al proceso de pruebas será contener todas las prácticas relativas a las pruebas.

CMMI [5][6] es modelo de referencia más difundido dentro de la industria del software, por tanto si queremos que la mejora del proceso de pruebas se produzca de forma paralela a la del resto de los procesos de la organización será necesario que el modelo de referencia sea compatible e integrable con CMMI.

Actualmente, los modelos de referencia relacionados con la mejora del proceso de pruebas más destacables son: TMM [7], TMMI [8][9], TPI [10] y TMap [11][12]. Con el objetivo de evaluar si alguno de ellos proporciona una solución válida al problema definido anteriormente, se ha realizado un análisis de cada uno de ellos. Además de los modelos citados, se va a incluir CMMI en el análisis ya que los autores consideran adecuado evaluar si este modelo proporciona el soporte adecuado para llevar a cabo la mejora del proceso de pruebas.

CMMI for Development (*Capability Maturity Model Integration for Development*) es un modelo de madurez de mejora de procesos para el desarrollo de productos y servicios. En relación al proceso de pruebas define 3 áreas de proceso: *Product Integration*, *Validation* y *Verification*; sin embargo no cubren todas las necesidades del proceso de pruebas. El grado de abstracción con respecto a este proceso es muy elevado, lo trata como algo genérico mientras que necesita de una atención más centrada y una definición más exhaustiva.

TMM (*Test Maturity Model*) es un modelo de referencia centrado en el Proceso de Pruebas. La mejora del proceso de pruebas es soportada por único área de proceso que define un conjunto de niveles y objetivos de madurez en el TMM. Propone 5 niveles de madurez y describe una estructura interna implementada en cada uno de ellos, tal y como se ve en la figura 1.

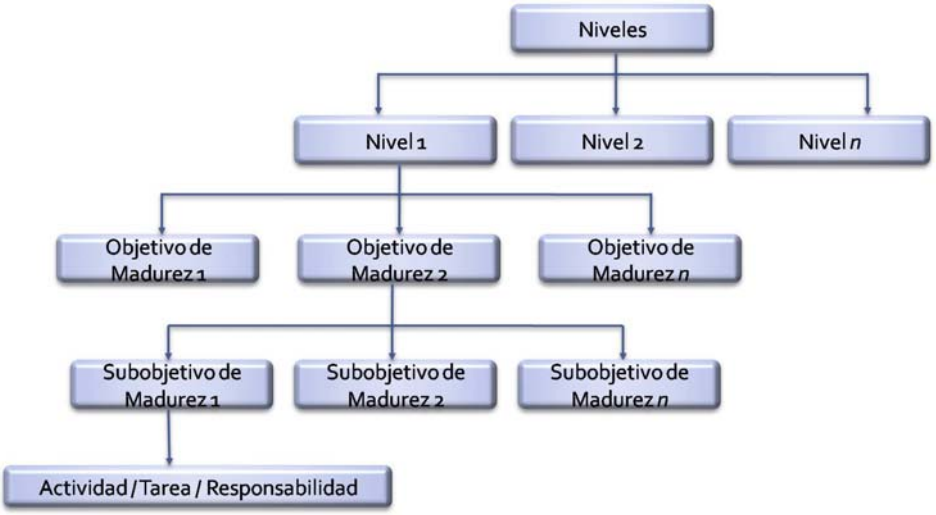

Figura 1. Estructura de TMM.

TMMi (*Test Maturity Model Integration*) es un modelo de referencia dirigido al proceso de pruebas. TMMi utiliza el concepto de niveles de madurez para la evaluación y mejora del proceso. Define cinco niveles de madurez con sus correspondientes áreas de proceso, así como los objetivos específicos y genéricos y prácticas específicas y genéricas. La figura 2 muestra su estructura.

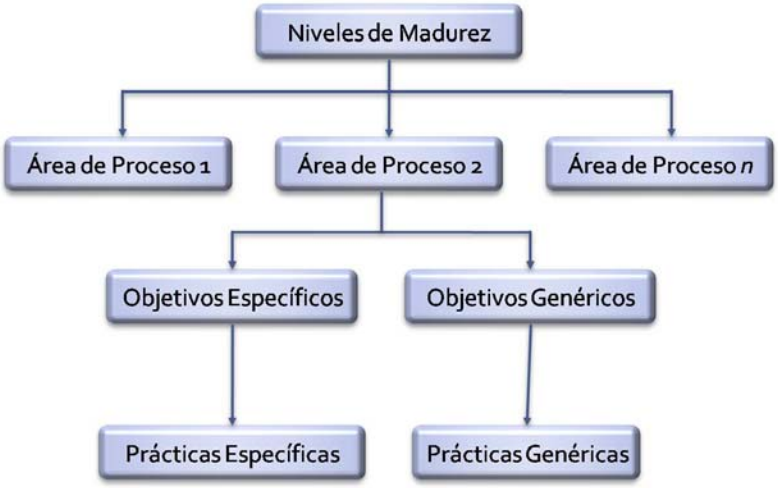

Figura 2. Estructura de TMMi

TPI (*Test Process Improvement*) es un modelo de referencia dirigido al proceso de pruebas. Propone un conjunto de 20 áreas clave con diferentes niveles de madurez. Define una Matriz de Madurez de Pruebas para establecer las relaciones existentes entre las diferentes áreas, y determinar la relevancia de cada uno de los niveles. Además, establece un conjunto de Puntos de Comprobación y Sugerencias de Mejora.

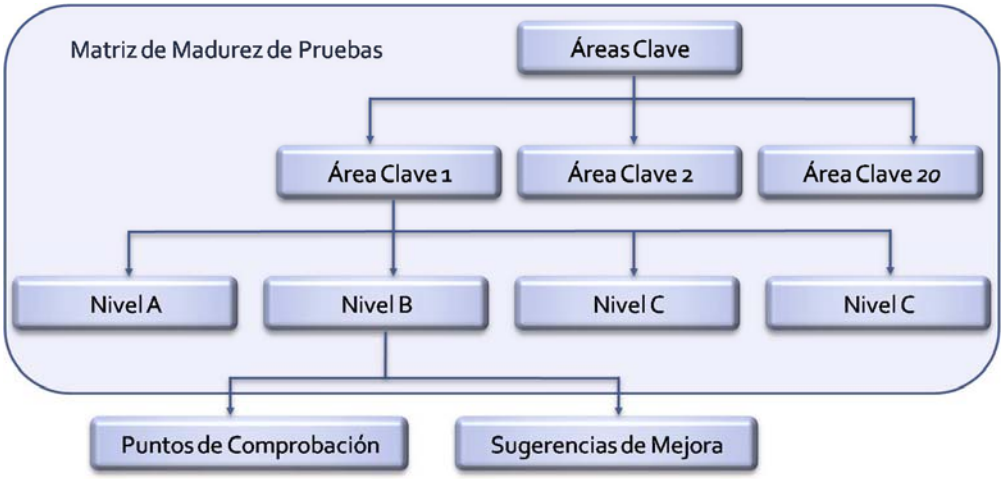

Figura 3. Estructura de TPI.

TMap (*Test Management approach*) es una metodología para el desarrollo de pruebas software. Los cuatro aspectos fundamentales de TMap son los siguientes:

1. TMap está basado en un enfoque BDMT (*Business Driven Test Management*).
2. TMap describe un proceso de pruebas estructurado.
3. TMap proporciona una caja de herramientas: técnicas, infraestructura y organización; para la realización de las pruebas.
4. TMap es una metodología adaptativa.

A diferencia de los modelos presentados anteriormente, TMap no proporciona un marco de referencia para establecer el nivel de capacidad o madurez del proceso de pruebas, sino que es una metodología que define cómo organizar, planificar, ejecutar y controlar las pruebas para obtener un producto de calidad. A pesar de ello, los autores han considerado apropiado incluirlo en este estudio.

Para realizar el análisis de los modelos citados se han definido seis criterios: Tipo, Cobertura completa, Completamente definido, Representación por étapas, Representación continua, y Conformidad con CMMI. Éstos criterios se encuentran definidos en la Tabla 1.

Criterio	**Definición**
Tipo	Tipo de modelo de mejora. Puede ser un modelo de mejora de procesos si cubre los diferentes procesos existentes en la organización, o un modelo de mejora del proceso de pruebas si solamente cubre el proceso de pruebas.
Cobertura Completa	Un modelo de referencia proporciona Cobertura Completa si incluye todas las prácticas relativas a las pruebas.
Completamente Definido	Un modelo de referencia está Completamente Definido si todas las prácticas definidas en el modelo están completamente detalladas.
Representación por etapas	Un modelo de referencia proporciona la Representación por etapas si describe todos los objetivos y prácticas necesarias para establecer el nivel de madurez de la organización.

Criterio	Definición
Representación continua	Un modelo de referencia proporciona la Representación continua si describe todos los objetivos y prácticas necesarias para establecer el nivel de capacidad de la organización.
Conformidad con CMMI	Un modelo de referencia está en Conformidad con CMMI si tiene una estructura compatible con la estructura de CMMI y la integración entre ambos modelos puede realizarse de forma completa y sencilla.

Tabla 1. Criterios definidos para realizar el análisis de los modelos de referencia.

La Tabla 2 resume los resultados obtenidos del estudio que ha sido realizado sobre los modelos citados anteriormente y teniendo en cuenta los criterios definidos.

Criterio/Modelo	CMMI	TMM	TMMi	TPI	TMap
Tipo	Modelo de mejora de procesos	Modelo de mejora del proceso de pruebas	Modelo de mejora del proceso de pruebas	Modelo de mejora del proceso de pruebas	Modelo de mejora del proceso de pruebas
Cobertura Completa	No	Si	Si	Si	Si
Completamente Definido	Si	Si	No	Si	Si
Representación por etapas	Si	Si	Si	Si	No
Representación continua	Si	No	No	No	No
Conformidad con CMMI	Si	No	No	No	No

Tabla 2. Resultados del análisis de los modelos de mejora del proceso de pruebas.

Tras el análisis de los resultados, los autores han determinado que ninguno de los modelos existentes soluciona el problema descrito anteriormente: mejorar el proceso de pruebas mientras se realiza la mejora de otros procesos de la organización. La siguiente sección propone una nueva área de proceso cuyo objetivo es mejorar el proceso de pruebas.

3. TestPAI – Un área de proceso de pruebas integrado con CMMI

TestPAI es un área de proceso de pruebas integrado con CMMI. Se encuentra situada en el nivel 3 con los procesos de ingeniería. TestPAI ha sido desarrollado para proporcionar el marco de trabajo necesario para que la mejora del proceso de pruebas se desarrolle de forma paralela a la mejora de otros procesos implementados en la organización.

Cada vez hay más pequeñas y medianas empresas que están adquiriendo conciencia a cerca de la importancia del proceso de prueba y la mejora del mismo. Muchas de ellas están demandando un marco de trabajo que les proporcione las herramientas necesarias para realizar una mejora del proceso de pruebas de un modo

poco costoso y sencillo. Además, muchas de esas organizaciones se encuentran trabajando con CMMI para mejorar otros procesos implementados, por lo que necesitan que dicho marco de trabajo sea totalmente compatible e integrable con CMMI. Así, TestPAI surge para proporcionar una solución a este problema.

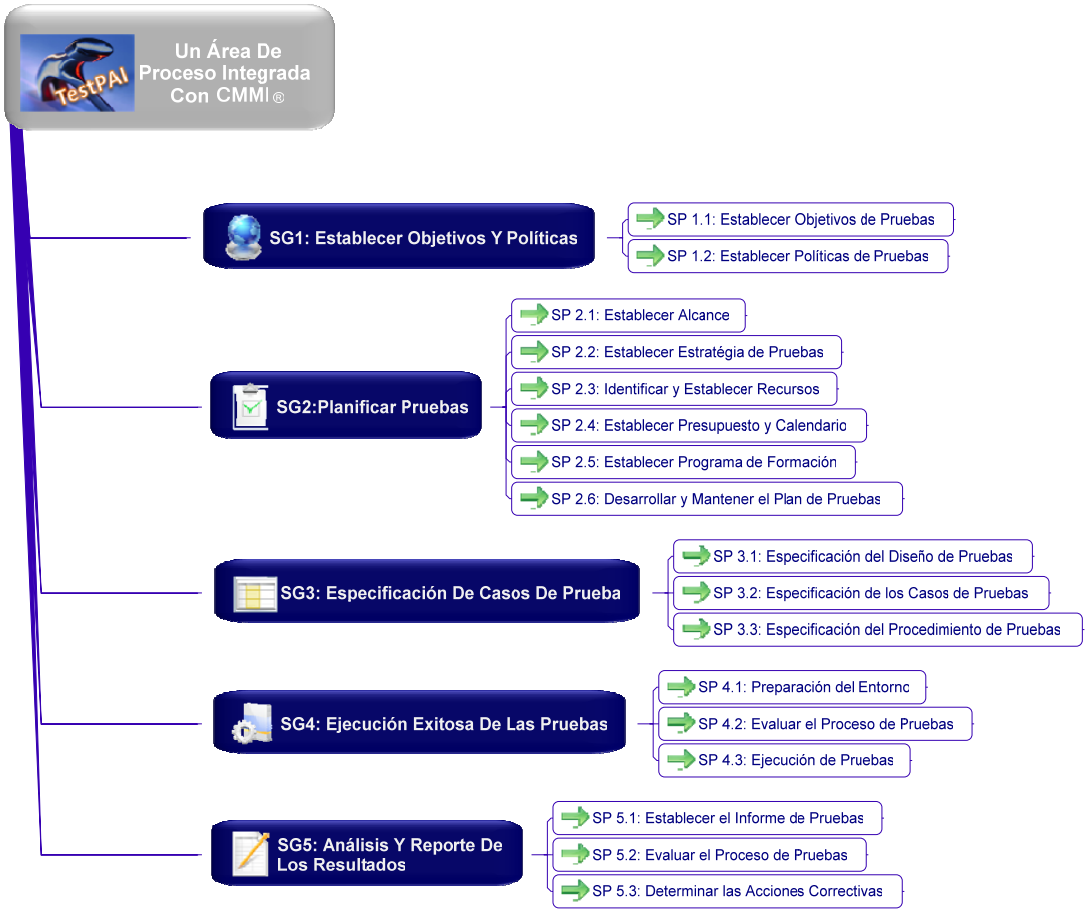

Figura 4. Objetivos y Prácticas Específicas de TestPAI

TestPAI incluye y define todas las prácticas relativas a las pruebas. Además, tiene la misma estructura que CMMI y contempla tanto la representación continua como la representación por etapas. Todo esto permite una sencilla y completa integración entre TestPAI y CMMI. TestPAI define cinco objetivos específicos (SG) y sus correspondientes prácticas específicas (SP), las cuales se muestran en la Figura 4.

El propósito del primer objetivo específico es definir los objetivos y políticas asociados a las pruebas. Se establecerán los objetivos de pruebas y la política de pruebas que permita alcanzar los objetivos definidos. La figura 5 ilustra este objetivo.

Figura 5. Prácticas Específicas del SG1.

El propósito del segundo objetivo específico es desarrollar y mantener el Plan de Pruebas. La planificación implica establecer y analizar los diferentes elementos que afectan a la gestión y desarrollo de las pruebas tales como definir alcance, analizar riegos, definir la estrategia o asignar recursos entre otros. La Figura 6 muestra las seis prácticas específicas que contiene éste objetivo.

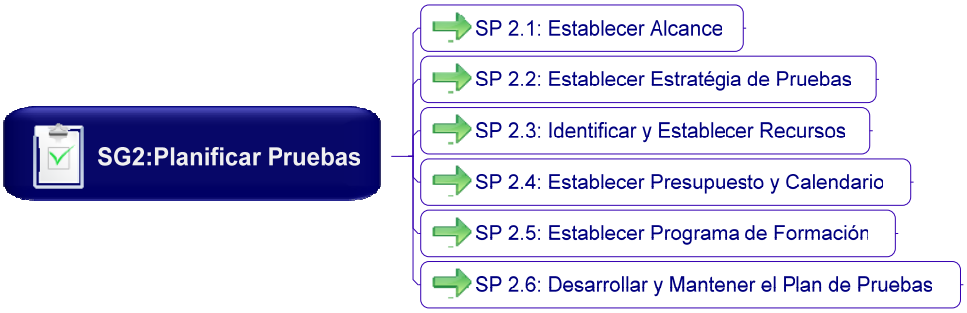

Figura 6. Prácticas Específicas del SG2.

El propósito del tercer objetivo específico es establecer la información necesaria para realizar una ejecución efectiva de las pruebas. Un proceso de pruebas bien definido implica determinar características que no se encuentran descritas en el Plan de Pruebas, como pueden ser las entradas, salidas o resultados esperados de una prueba. La figura 7 muestra las prácticas específicas de este objetivo específico.

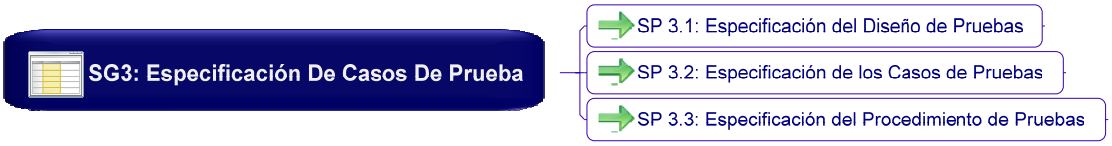

Figura 7. Prácticas Específicas del SG3

El propósito del cuarto objetivo es proporcionar la infraestructura adecuada para realizar las pruebas. Una ejecución de pruebas satisfactoria es un elemento importante en la mejora de la calidad del software, y preparar la infraestructura correctamente contribuye a que la ejecución se desarrolle de forma disciplinada. La figura 8 muestra las prácticas específicas de este objetivo específico.

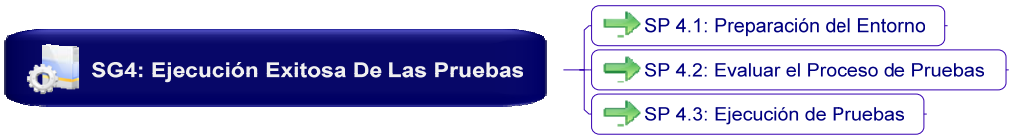

Figura 8. Prácticas Específicas del SG4

El propósito del quinto objetivo es es analizar, evaluar y comunicar los resultados obtenidos, con el objetivo de poder evaluar el proceso de pruebas y la calidad del producto. Contiene tres prácticas específicas que se muestran en la Figura 9.

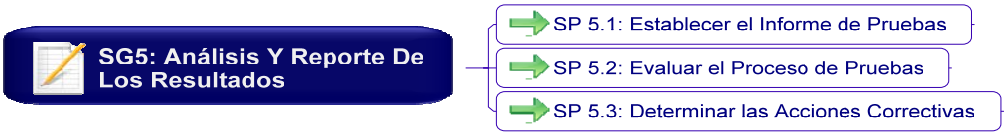

Figura 9. Prácticas Específicas del SG5.

En este apartado se ha descrito brevemente el conjunto de objetivos específicos definidos en TestPAI. La versión completa de TestPAI está disponible para ser consultada en la dirección http://sel.inf.uc3m.es/asanz/testpai/testpai_pa_fullversion_esp.pdf.

4. Implementación de TestPAI

TestPAI ha sido implementado en una organización Española con el objetivo de verificar que se produce la mejora del proceso de pruebas, mediante una integración sencilla y completa con CMMI.

La organización en la que se ha llevado a cabo la implementación es una de las principales compañías de seguros, cuyo mercado está dirigido al sector agrario. Tiene alrededor de 200 empleados, de los cuales aproximadamente 30 están en el departamento de informática.

Su misión es proporcionar las soluciones necesarias para el resto de las áreas de la organización con un alto nivel de calidad y optimizando el plazo de entrega para cada trabajo.

En una iniciativa previa, la organización llevó a cabo una mejora de procesos software que le permitió alcanzar un nivel 2 de CMMI. El objetivo de negocio actual es conseguir un nivel 3 de CMMI, pero además también se encuentran interesados en mejorar su proceso de pruebas. Desde que lograron un nivel 2 de CMMI los beneficios de la organización se han visto incrementados por lo que no desean cambiar de filosofía. Por ello, necesitan un modelo de referencia que les permita mejorar el proceso de pruebas al mismo tiempo que se realiza la mejora del resto de los procesos implementados en la organización. Para resolver su problema se propone la implementación de TestPAI.

Para poder llevar a cabo la implementación de TestPAI hubo que definir los diferentes pasos a seguir. En primer lugar, fue necesario Establecer los Objetivos de Mejora de la organización, en función de los antecedentes y los objetivos de la organización. Se detectó que existía una gran cantidad de mantenimiento correctivo, lo que denotaba que las pruebas eran ineficientes. A continuación, se llevó a cabo un Estudio de la Práctica Actual con el propósito de determinar cuáles eran las buenas prácticas y cuáles eran las malas prácticas que se estaban llevando a cabo en la organización. Para ello, se desarrollaron entrevistas tanto individuales (al personal clave, jefe de proyecto, analistas, etc.) como en grupo a los equipos de trabajo. Después de esto, se realizó una Adaptación de los Procesos de desarrollo de la organización para incluir las prácticas propuestas por TestPAI. Fue necesario definir nuevos procesos y modificar algunos ya existentes, además se llevó a cabo un proceso de verificación de los mismos. Finalmente, se realizó la Evaluación del Uso y la Mejora del Proceso de Pruebas Definido. Los equipos de trabajo comenzaron a utilizar los procesos en sus nuevos proyectos; al finalizar cada una de sus actividades rellenaban una lista de verificación que permitía evaluar si se había seguido el proceso y si se había seguido bien. Además, se consigue que el personal se implique en la actividad de mejora continua, y el refinamiento del proceso inicial aumentando la flexibilidad y adaptabilidad del proceso. La Figura 10 (ver apéndice) muestra estos pasos y las principales actividades de cada uno de ellos.

También se realizó la evaluación de la implementación de TestPAI mediante un proceso de verificación que se encuentra basado en la creación y uso de listas de comprobación. Este proceso de verificación define dos tipos de verificación:

A. Verificación de la realización de los elementos de verificación (EV): se comprueba que los EV que el responsable del proyecto ha notificado como realizados, realmente han sido realizados. Este tipo de verificación podrá realizarse de forma automática o manual.
B. Verificación de la calidad en la realización de los EV: se evaluará objetivamente la calidad en la realización de los EV.

La Figura 11 (ver apéndice) resume el proceso de verificación que se ha realizado, y la Tabla 3 (ver apéndice) muestra un ejemplo ilustrativo de un fragmento de una lista de comprobación que ha sido utilizada en el proceso de verificación de la implementación de TestPAI.

5. Conclusiones

TestPAI es un área de proceso definido y detallado que puede integrarse totalmente con CMMI. Esta área de proceso permite llevar a cabo la mejora del proceso de pruebas mientras se está realizando la mejora de otros procesos implementados dentro de la organización.

Muchas organizaciones Españolas utilizan CMMI para mejorar sus procesos. En muchos casos, el proceso de pruebas juega un papel importante dentro de su actividad de negocio, por lo que se encuentran interesadas en realizar la mejora del mismo. CMMI no proporciona un soporte completo para desarrollar actividades de mejora del proceso de pruebas, por lo que no es un modelo de referencia adecuado para este propósito. TestPAI es un área de proceso dedicada a resolver este problema; es compatible e integrable con CMMI, tanto en su representación continua como en su representación por etapas.

TestPAI ha sido implementada en una organización real. Para poder llevar a cabo dicha implementación, fue necesario definir un nuevo proceso de pruebas y adaptar otros procesos ya existentes en la organización como son el proceso de diseño y el proceso de construcción.

La implementación se llevó a cabo en varios proyectos piloto, obteniendo resultados similares en todos ellos. Es necesario que el personal realice un programa de formación previa en las actividades de pruebas y en las herramientas tecnológicas que ofrecen soporte a las mismas. Además, se debe mejorar las habilidades individuales y competencias del personal involucrado en las actividades de pruebas.

Agradecimientos

Este trabajo está soportado por el proyecto *Buen Gobierno de las Tecnologías de la Información* financiado por ATOS-ORIGIN S.A.

Referencias

[1] Kit, E. *Software Testing in the Real World*. Addison-Wesley. 1995.

[2] García, A. de Amescua, M. Velasco y A. Sanz, “Ten Factors that Impede Improvement of Verification and Validation Processes in Software Intensive Organizations”, *Software Process Improvement and Practice,* vol. 13, nº 4, pp 335-343, 2008.

[3] Software Engineering Institute,*CMMI Maturity Profile March 2008 Report*. Carnegie Mellon University, (http://www.sei.cmu.edu/appraisal-program/profile/pdf/CMMI/2008MarCMMI.pdf), 2008.

[4] Bush, M. y Dunaway, D. *CMMI Assessments. Motivating Positive Change*. Addison-Wesley. 2005

[5] SEI, *CMMi® for Development*. SEI, Carnegie Mellon University, 2006.

[6] Paulk, M., Weber, C., Curtis, B. y Chrisis, M. *The Capability Maturity Model.* Addison-Wesley. 1995.

[7] Burnstein, I. *Practical Software Testing*. Springer-Verlag. 2002.

[8] van Veenendaal, E., *Guidelines for Testing Maturity*. Published in: STEN Journal, Vol IV, 2006.

[9] van Veenendaal, E., *Test Maturity Model Integration (TMMi) Versión 1.0*. TMMI Foundation (www.tmmifoundation.org), 2008.

[10] Koomen, T. *Test process improvement: a practical step-by-step guide to structured testing*. Addison Wesley. 1999.

[11] Pol, M., Teunissen,R. y van Veenendaal, E. *Software Testing. A guide to the TMap Approach*. Addison-Wesley. 2002.

[12] Koomen, T., van der Aalst, L., Broekman, B. y Vroon, M. *TMap Next for result-driven testing*. UTN Publishers. 2006.

Apéndice

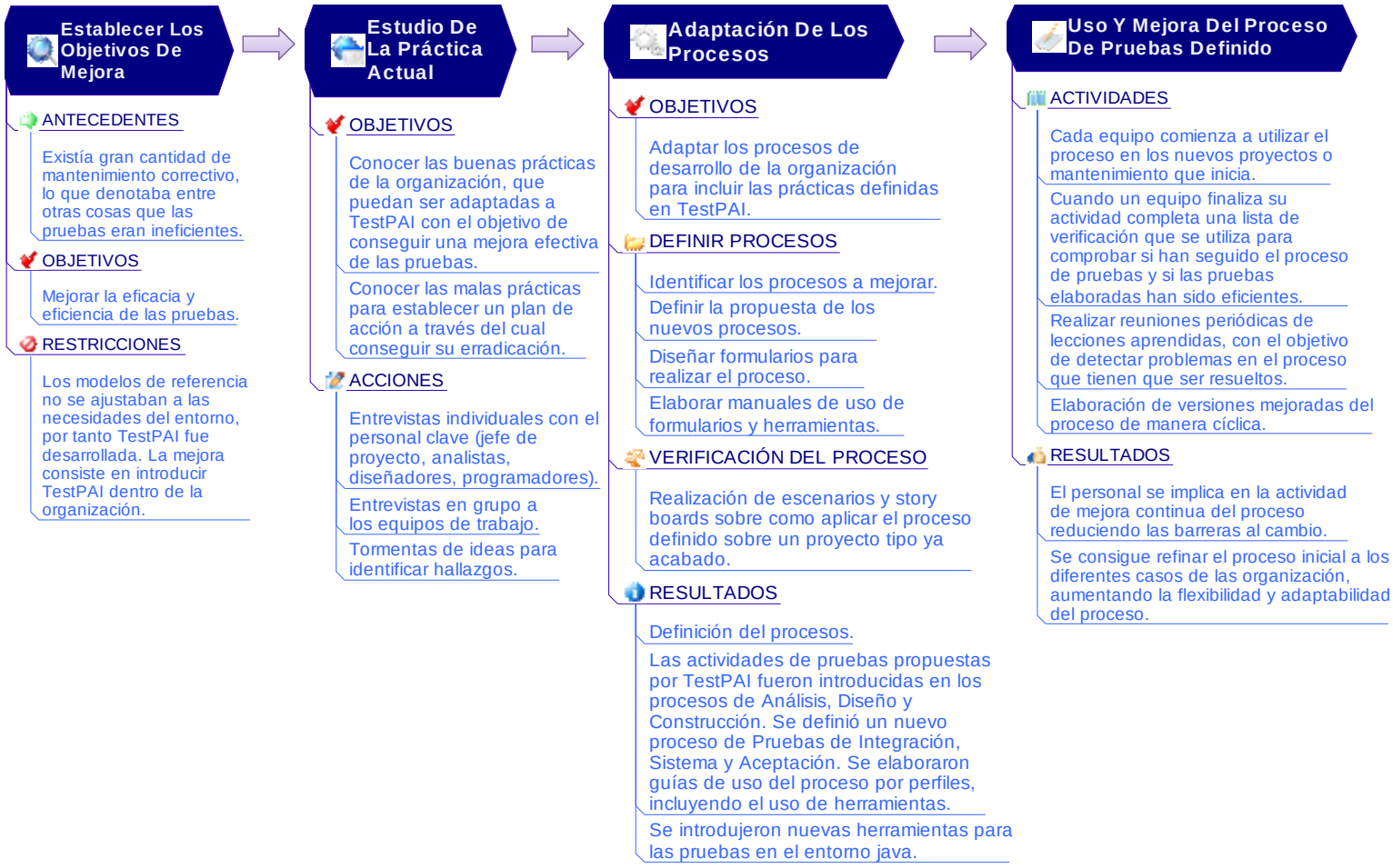

Figura 10. Implementación de TestPAI.

Figura 11. Proceso de Verificación

Responsable:	Nº Trabajo:	Tipo Trabajo:		
Entorno de Desarrollo:		Fecha:		
ELEMENTOS A VERIFICAR				
ELEMENTO	ACEPTACION	OBSERVACIONES	ACCIONES CORRECTIVAS	FECHA DE CIERRE
Se han registrado los resultados de la ejecución de las pruebas de sistema.	☐			
Se han evaluado los resultados de la ejecución de las pruebas de sistema.	☐			
Se han aceptado los resultados de la ejecución de las pruebas de sistema.	☐			
…..	☐			

Tabla 3. Ejemplo de un fragmento de una Lista de Verificación.

Propuesta para pruebas dirigidas por modelos usando el perfil de pruebas de UML 2.0

Beatriz Pérez Lamancha
Centro de Ensayos de Software (CES), Instituto de Computación,
Universidad de la República de Uruguay
bperez@fing.edu.uy

Pedro Reales Mateo, Ignacio García-Rodriguez de Guzmán, Macario Polo Usaola
Grupo de Investigación ALARCOS, Departamento de Tecnologías y Sistemas de Información, Universidad de Castilla-La Mancha, España
{pedro.reales,ignacio.grodriguez,macario.polo}@uclm.es

Abstract

This work presents a proposal for testing in Model Driven Engineering environment. The system design models represented in UML are automatically transformed in testing models conforms with the UML Testing Profile. For automatically generate the test cases, an extension for sequence diagram in UML is defined where pre and pos condition are attached to the models. This information is used later for the test oracle generation. The extension uses OCL to define the pre and pos conditions to the test cases.

Key words: model based testing, UML 2.0 testing profile, oracle generation

Resumen

Se 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 metamodelo 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 postcondiciones en el diagrama.

1. Introducción

El perfil de pruebas para UML (*UML Testing Profile*) define un lenguaje para diseñar, visualizar, especificar, analizar, construir y documentar los artefactos de un sistema de pruebas. Extiende UML con los conceptos específicos de pruebas, se basa en el metamodelo de UML y reutiliza su sintaxis definiendo conceptos para: observación del comportamiento de las pruebas y las actividades durante las pruebas, arquitectura de las pruebas, datos de pruebas y tiempo [5].

En este artículo se presenta una propuesta para la generación automática de casos de prueba en el contexto de Ingeniería dirigida por modelos (*Model Driven Engineering*). La metodología se basa en el metamodelo de UML y el perfil de pruebas de UML, realizando transformaciones desde los modelos UML al modelo de pruebas, utilizando como modelo de descripción de comportamiento del sistema el diagrama de secuencias de UML. Para la realización de transformaciones se utiliza el estándar de OMG QVT (Query/View/Transformation).

Dado que para las pruebas dirigidas por modelos es esencial la automatización de todo el proceso, uno de los problemas más importantes al que hay que enfrentarse es la generación automática de los oráculos de las pruebas ya que éstos son dependientes del dominio de la aplicación. Este hecho hace necesario añadir información adicional a los modelos que definen el diseño del sistema para poder generar los oráculos. Se presenta una extensión al metamodelo del diagrama de secuencia de UML para representar la información dependiente del dominio como pre y postcondiciones anotadas usando OCL que servirá posteriormente para generar los oráculos de las pruebas. Los diagramas de secuencia extendidos con pre y postcondiciones son transformados luego en modelos de prueba que son instancias del perfil de pruebas de UML.

El artículo se estructura de la siguiente manera: sección 2 presenta los principales trabajos relacionados en los que se basa esta investigación, en la sección 3 se describe la propuesta para pruebas dirigidas por modelos usando el perfil de pruebas de UML, en la sección 4 se muestra un ejemplo práctico de dicha metodología y por último en la sección 5 se presentan las conclusiones y trabajo a futuro.

2. Trabajos relacionados

En esta sección se describen los principales trabajos relacionados con nuestra investigación que se centra en la definición automática de un oráculo para las pruebas, derivando pruebas en el contexto de la ingeniería dirigida por modelos y generando un modelo de pruebas basado en el perfil de pruebas de UML.

2.1. Oráculo para las pruebas

El oráculo es el mecanismo de que se dota a cada caso de prueba para determinar, tras su ejecución, si el sistema que se está probando supera o no el caso. No existe un mecanismo

que permita describirlo de manera genérica y, en la práctica, se implementa siempre manualmente [6]. Una de las principales dificultades con las que se encuentra la investigación en el área del *testing*, y que, de acuerdo con Bertolino [7], representa un gran obstáculo para avanzar en su automatización, es la descripción del oráculo. Bertolino hace referencia al oráculo ideal, que describe como "un método mágico [sic] que proporciona las salidas para cada caso de prueba, aunque en la práctica se implemente como un motor o heurística que emite un veredicto de paso o fallo sobre las salidas observadas".

El trabajo de Baresi y Young, del año 2001 [6], presenta el estado del arte en cuanto al problema del oráculo. La mayor parte de las propuesta analizadas se refieren a la inserción de instrucciones en los programas que realizan algún de comprobación, como las assert de Java, macros de C o extensiones para lenguajes que permiten, mediante algún tipo de preprocesamiento, embeber en el código la comprobación de restricciones.

En este trabajo definimos una forma de anotar la información requerida para poder construir el oráculo de las pruebas desde los modelos de diseño del sistema, en particular, se extiende el metamodelo de UML para definir dicha información en los diagramas de secuencia del sistema.

2.2. Perfil de Pruebas de UML (UML Testing Profile)

El perfil de pruebas de UML (UML 2.0 Testing Profile, U2TP) define un lenguaje para diseñar, visualizar, especificar, analizar, construir y documentar los artefactos de prueba del sistema. Extiende UML 2.0 con conceptos específicos para las pruebas, dichos conceptos son agrupados en: arquitectura de pruebas, datos de prueba, comportamiento de las pruebas y tiempo de prueba. Al ser un perfil, se integra perfectamente con UML y se basa en el metamodelo de UML, reutilizando su sintaxis [8].

La arquitectura de las pruebas es la definición de todos los conceptos necesarios para realizar las pruebas, se define en el paquete de la arquitectura el contexto de las pruebas y los conceptos necesarios para definir las pruebas. El contexto de prueba (*Test Context*) permite agrupar casos de prueba para describir una configuración de pruebas y definir el control de las pruebas, o sea, el orden requerido para la ejecución de los casos de prueba. La configuración de las pruebas (*Test configuration*) muestra la estructura de comunicación entre los componentes de prueba y el sistema bajo prueba (*system under test ó SUT*). Los componentes de prueba (*Test components*), que son definidos como objetos

dentro del sistema de prueba y pueden comunicarse con el SUT o los otros componentes para llevar a cabo el comportamiento de la prueba.

El comportamiento de la prueba especifica las acciones y evaluaciones necesarias para probar el objetivo de prueba (*test objetive*) que describe qué debe ser probado. El caso de prueba (*test case)* es una operación de un contexto de prueba que especifica cómo un conjunto de componentes cooperan con el SUT para realizar el objetivo de prueba. Un veredicto (*verdict*) es una enumeración predefinida que especifica los posibles resultados de las pruebas (pasa, falla, error, inconclusa), las acciones de validación (*validation action*) son realizadas por el componente de prueba para indicar que el arbitro (*arbiter*) esta informado de los resultados de las pruebas. El árbitro evalúa los resultados de las pruebas realizadas por los distintos objetos dentro del sistema de prueba para determinar un veredicto global para el caso de prueba o contexto. Los tiempos (*timers*) son usados para manipular y controlar el comportamiento de las pruebas y asegurarse que los casos de prueba terminan [8]. Otro aspecto importante de las especificación es la definición y codificación de los datos de prueba, para esto U2TP soporta *wildcards*, conjuntos de datos (*data pools*), particiones de datos (*data partitions*), selectores (*data selectors*) y reglas de codificación [8].

2.3. Ingeniería dirigida por modelos (Model Driven Engineering)

La ingeniería dirigida por modelos (Model-Driven Engineering ó MDE) combina: (i) Lenguajes de modelado específicos del dominio (Domain-specific modeling languages ó DSML), que son descritos usando metamodelos que definen las relaciones entre los conceptos en un dominio y especifican en forma precisa la semántica y las restricciones asociadas con esos conceptos del dominio; (ii) motores de transformación y generadores que analizan ciertos aspectos de los modelos y que sintetizan varios tipos de artefactos, tales como código fuente, simulación de entradas, descripciones de distribución XML o representaciones alternativas de modelado [1]. Estándares tales como *Query/Views/Transformations (QVT)* y *Meta Object Facility(MOF)* han sido definidos como parte del estándar de OMG de Arquitectura dirigida por modelos *(Model-Driven Architecture ó MDA)*, los cuales pueden ser utilizados como la base para herramientas MDE específicas del dominio[1].

El enfoque MDA utiliza la idea de separar la especificación del sistema de los detalles en que el sistema utiliza las capacidades de su plataforma. MDA provee un enfoque para especificar un sistema independiente de la plataforma que lo soporta, especificar plataformas, elegir una plataforma particular para el sistema y transformar la especificación del sistema en una de las plataformas particulares [2]. MDA especifica tres vistas del sistema: 1) Vista independiente de la computación (CIM): focaliza en los requisitos del sistema, los detalles de estructura y procesamiento del sistema no son tenidos en cuenta. 2) Vista independiente de la plataforma (PIM): focaliza en la operación del sistema sin tener en cuenta detalles de una plataforma particular. 3) Vista específica de la plataforma (PSM): Combina la vista independiente de la plataforma con el objetivo en los detalles de su uso en un plataforma específica [2].

La transformación de modelos es el proceso de convertir un modelo en otro para el mismo sistema. Se utiliza un lenguaje para describir dicha transformación [2]. Una transformación establece un conjunto de reglas que describen cómo un modelo expresado en un lenguaje origen puede ser transformado en un modelo en un lenguaje destino. Las categorías principales de transformaciones en MDA son los refinamientos (mappings) verticales y horizontales. Los refinamientos verticales relacionan modelos del sistema situados en distintos niveles de abstracción, como transformaciones de PIM a PSM ó de PSM a PIM. Los refinamientos horizontales, relacionan o integran modelos que cubren distintos aspectos en un mismo nivel de abstracción, mantienen la consistencia entre distintos niveles y garantizan que la información modelada sobre una entidad del sistema es consistente con lo que se dice sobre ella en cualquier otra especificación situada en el mismo nivel de abstracción [3].

Para realizar las transformaciones, OMG ha definido un estándar para la definición de transformaciones llamado *QVT (Query/View/Transformation)* [4].

2.4. Pruebas dirigidas por modelos (Model Driven Testing)

Puede aplicarse el perfil de pruebas de UML a un modelo diseñado con UML para derivar un modelo de pruebas. Los niveles de abstracciones de MDA pueden aplicarse al modelado de las pruebas, los modelos de pruebas pueden ser independientes de la plataforma o específicos de la plataforma en forma previa a generar el código de pruebas ejecutable [9]. En la Figura 1(a) se muestra el enfoque propuesto por Dai siguiendo el

enfoque MDA para los modelos de pruebas. En la Figura 1(b) se muestran las transformaciones para las pruebas basadas en los metamodelos. El metamodelo fuente es el de UML y el metamodelo destino es el metamodelo de UML Testing Profile (U2TP). La transformación entre modelos es especificada por reglas de acuerdo a la especificación QVT [9]. Esta propuesta es teórica y no ha sido desarrollada.

En el libro de Baker [8] se definen modelos de prueba utilizando el perfil de pruebas de UML, pero las transformaciones se realizan manualmente, sin usar lenguaje específico.

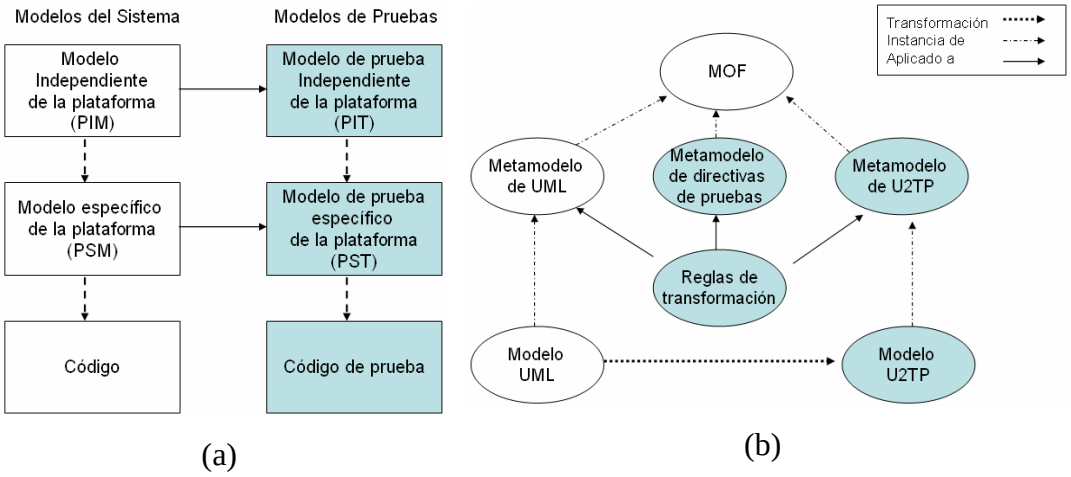

Figura 1. Transformación de modelos de pruebas[9]

3. Propuesta para pruebas dirigidas por modelos usando el perfil de pruebas de UML

Para definir nuestro método de derivación de pruebas dirigido por modelos, nos basamos en la propuesta de Dai [9] (ver sección 2.3).

En las pruebas dirigidas por modelos resulta imprescindible poder generar casos de prueba completos en forma automática. Esto implica representar la información dependiente del dominio para las pruebas en los modelos UML y poder transformarla luego en el oráculo de pruebas en los distintos modelos de prueba.

La propuesta presentada en este artículo añade una pequeña extensión al metamodelo de UML (ver Figura 2) para agregar la información requerida para el oráculo de las pruebas en los diagramas de secuencia. Dado que un diagrama de secuencia se corresponde con un escenario relevante que debe ser probado, el diagrama se anota de forma que incluya información sobre el resultado esperado como resultado de la ejecución de dicho escenario.

En la Figura 2 se muestra la extensión del metamodelo de UML definida, donde en la clase *InteractionFragment* se le agrega una *Constraint* de OCL, y dos relaciones, una que representa la precondición del diagrama de secuencia y otra que representa sus postcondiciones. Se agrega también una restricción que indica que dichas pre y postcondiciones no se aplican al *InteractionFragment* de tipo *StateInvariant*, ya que éste representa invariantes donde las condiciones no cambian.

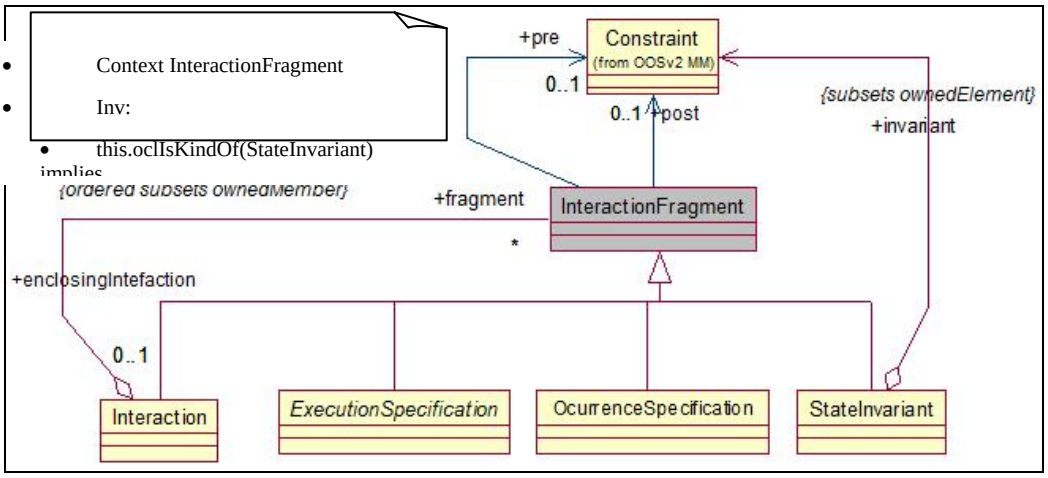

Figura 2. Extensión al metamodelo de UML para representar pre y postcondiciones para las pruebas

Un *InteractionFragment* es un trozo de una interacción, por lo que la semántica de nuestra extensión es que cualquier segmento de una interacción puede tener una pre y poscondición a ser utilizada como oráculo para las pruebas. Esto nos permite poder derivar casos de prueba a distintos niveles: unitario, de integración ó funcionales.

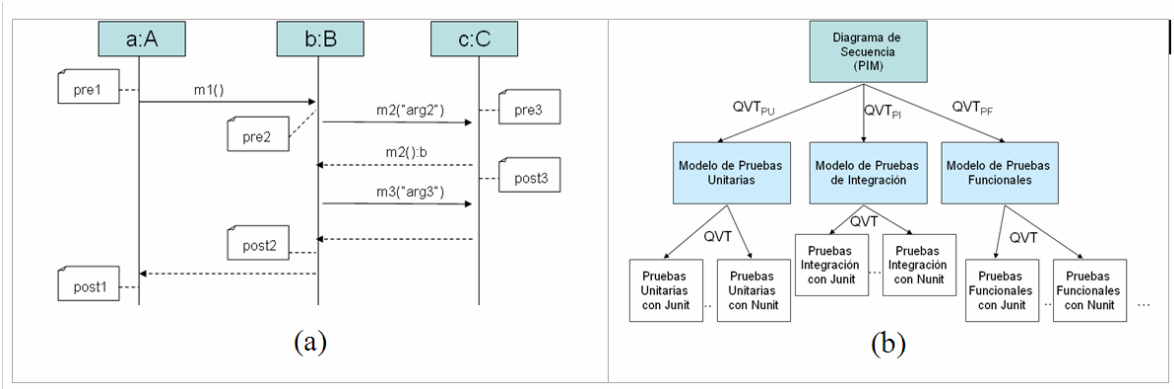

Figura 3. Información para el oráculo según el nivel de prueba

En la Figura 3 (a) se pueden observar los distintos puntos donde se pueden poner pre y postcondiciones. Por ejemplo, la pareja pre1 y pos 1 tienen la información necesaria para construir el oráculo para todo el diagrama, lo que se conoce como una prueba funcional. La

pareja pre2 y pos2 dan la información necesaria para construir el oráculo para probar la interacción entre B y C, mientras que la pareja pre3 y pos3 nos permiten realizar pruebas unitarias de C. La Figura 3 (b) muestra las transformaciones posibles desde el diagrama de secuencia extendido a pruebas unitarias, de integración y funcionales. Siguiendo un enfoque MDA, primero se obtiene el modelo de pruebas en el nivel deseado y luego dicho modelo es refinado según la plataforma, en caso de que el lenguaje sea Java, podríamos derivar casos de prueba en JUnit. Las transformaciones serán realizadas utilizando QVT.

Para poder anotar las pre y postcondiciones en OCL es necesario conocer la información sobre las clases del sistema. A partir del diagrama de secuencia y el diagrama de clases se pueden realizar las transformaciones al modelo de pruebas, como se muestra en la Figura 4.

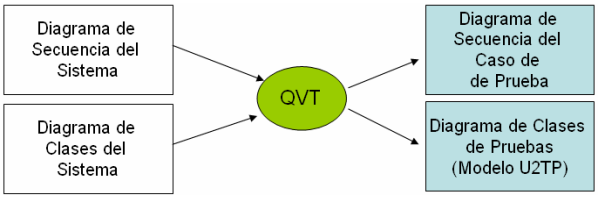

Figura 4. Transformaciones desde los diagramas de diseño del sistema al modelo de pruebas

3.1. Transformación del Diagrama de Secuencia extendido al Modelo de Pruebas

A partir del diagrama de clases, que representa la arquitectura del sistema, y de los diagramas de secuencia extendidos con pre y postcondiciones, que representan el comportamiento, se realizan las transformaciones para llegar al modelo de pruebas. Si bien como se explicó anteriormente podrían generarse distintos niveles de prueba, en esta sección nos enfocaremos en generar las transformaciones desde el diagrama de secuencia al modelo de pruebas funcionales.

El primer paso es construir la arquitectura del modelo de pruebas a partir de la arquitectura del sistema y de los modelos que representan el comportamiento (en nuestro caso, diagramas de secuencia). Los elementos de la arquitectura se exponen en la Tabla 1.

Elemento U2TP	Transformación realizada desde los modelos de clases y secuencia del sistema
TestContext	Para cada funcionalidad perteneciente al sistema aparece un TestContext con los casos de prueba de esa funcionalidad.
TestCase	Para cada diagrama de secuencia que representa un escenario de ejecución de la funcionalidad aparece un testCase. Se representa como una operación del TestContext.
TestObjective	Cada test case tendrá asociado el diagrama de secuencia que prueba.
Deployment	En cada TestContext aparecerá un Deployment, es la operación encargada de inicializar el sistema para las pruebas.
TestComponent	Por cada actor aparecerá un TestComponent que se encargará de interactuar con el SUT para realizar el test case.
Arbiter	Cada TestContext estará relacionado con un Arbiter. El Arbiter tendrá dos métodos, uno para almacenar resultados de casos de prueba y otro para leerlos.
DataPool	Cada TestContext estará relacionado con un DataPool.
Data Partition	Almacena datos relacionados con un *test case* y se relaciona con el *DataPool*
DataSelector	Aparecerá como una operación en un DataPool y habrá uno por cada testCase. Al ejecutarse deberá tener en cuenta la precondición asociada al diagrama de secuencia del sistema para generar los datos de prueba correctos para el caso de prueba al que esté ligado el DataSelector.

Tabla 1. Arquitectura del modelo de pruebas

Elemento U2TP	Transformación realizada desde los modelos de clases y secuencia del sistema
Diagrama de secuencia	Aparece un diagrama de secuencia nuevo por cada TestCase contenido en la arquitectura de las pruebas.
Línea de vida del TestContext	Cada diagrama de secuencias aparece una línea de vida que referencia a la clase TestContext que contiene el caso de prueba cuyo comportamiento está definiendo el diagrama.
Línea de vida del Arbiter	Cada diagrama de secuencias aparece una línea de vida que referencia a la clase Arbiter que dará el veredicto del test case.
Línea de vida del DataPool	Cada diagrama de secuencias aparece una línea de vida que referencia a la clase DataPool con los datos a usar para el test case.
Líneas de vida de los TestComponent	Cada diagrama de secuencias aparecen líneas de vida que se corresponden con los TestComponent que referencia los actores que intervienen en el test case.
Líneas de vida de los SUTs	Cada diagrama de secuencias aparecen líneas de vida que se corresponden con los SUTs, clases del sistema, que interactúan directamente con los actores.

Tabla 2. Comportamiento del modelo de pruebas (diagrama de secuencia de pruebas)

Una vez construida la arquitectura de sistema de pruebas, hay que generar el comportamiento asociado a cada uno de los casos de prueba. Para ello se generan una serie de diagramas de secuencia, de manera que cada uno modela el comportamiento de un caso de prueba. En la Tabla 2 se muestran los elementos que aparecen en cada uno de los diagramas de secuencia.

Una vez definidos los elementos que del diagrama de secuencia, se deben realizar las transformaciones para obtener los mensajes. Estos mensajes tienen un orden y, por tanto, las transformaciones que se muestran en la Tabla 3 han de realizarse en el orden especificado.

Elementos U2TP	Transformación realizada desde los modelos de clases y secuencia del sistema
Mensajes de inicio TestComponents	El TestContext le envía un mensaje a los TestComponents encargados de realizar el test case.
Mensaje para obtener los datos de prueba	Aparece un mensaje desde el TestContext hasta el DataPool o Data Partition invocando al DataSelector.
Mensajes de interacción con el SUT	Los mensajes que partían o llegaban a un actor, ahora parten o llegan al TestComponent que representa a ese actor hacia el SUT.
Mensajes de actualización del veredicto	Son los mensajes que van desde el TestComponent correspondiente hasta el Arbiter, para actualizar el Verdit del caso de prueba. Justo antes de ejecutar esta llamada cada TestComponent ha de calcular el Verdict correcto en función de la postcondición asociada al diagrama de secuencia del sistema.

Tabla 3. Mensajes del diagrama de secuencia de pruebas

3.1.1 Transformaciones en QVT

En QVT una transformación genera un modelo a partir de otro haciendo uso de relaciones. Por lo cual, la transformación en QVT que obtendrá el modelo de pruebas hará uso de relaciones, las cuales implementarán las diferentes transformaciones expuestas en las tablas anteriores.

A continuación se muestran las dos primeras transformaciones de la Tabla 1 diseñadas con QVT. Dada la complejidad del código QVT, se ha empleado la sintaxis gráfica propia del lenguaje para representar las transformaciones.

La relación QVT de Figura 5 genera los "*testContext*" necesarios para la arquitectura de pruebas, uno por cada funcionalidad. Mediante esta representación podemos

ver que la relación esta compuesta por dos dominios, uno de entrada (*useCase*), el cual tiene un conjunto de comportamientos asociados (*ownedBehavior*) y otro de salida (*classContext*). Así, para cada caso de uso del sistema se va a generar una clase estereotipada como <<TestContext>> que se encargará de probar la funcionalidad modelada. Además, en el campo "where" de la transformación se buscan todos los "*behaviors*" asociados al caso de uso que sean del subtipo "*Interaction*", es decir, que sean un diagrama de secuencias, y se ejecuta la transformación de generación de casos de prueba para generar, por cada diagrama de secuencias, un caso de prueba.

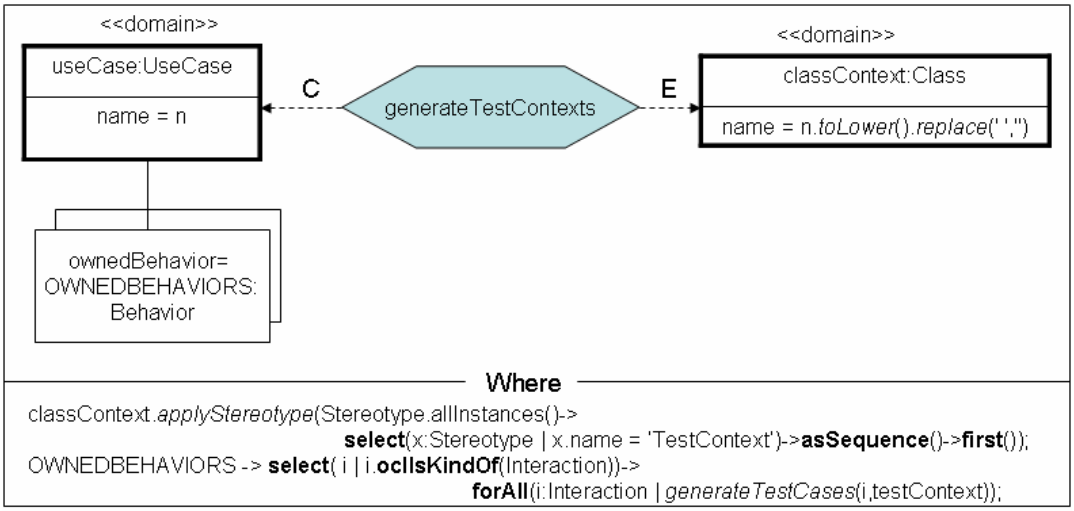

Figura 5 - Transformación para generar los diferentes "TestContext"

La Figura 6 muestra la representación visual de la relación QVT que implementa la segunda transformación de la Tabla 1, la cual se encarga de generar los test cases. Se puede ver que existe un dominio de entrada, un diagrama de secuencias, y dos dominios de salida, el "*testContext*" que contendrá el "*testCase*", el segundo dominio de salida, que ejecutará la prueba referente al diagrama de secuencias de entrada.

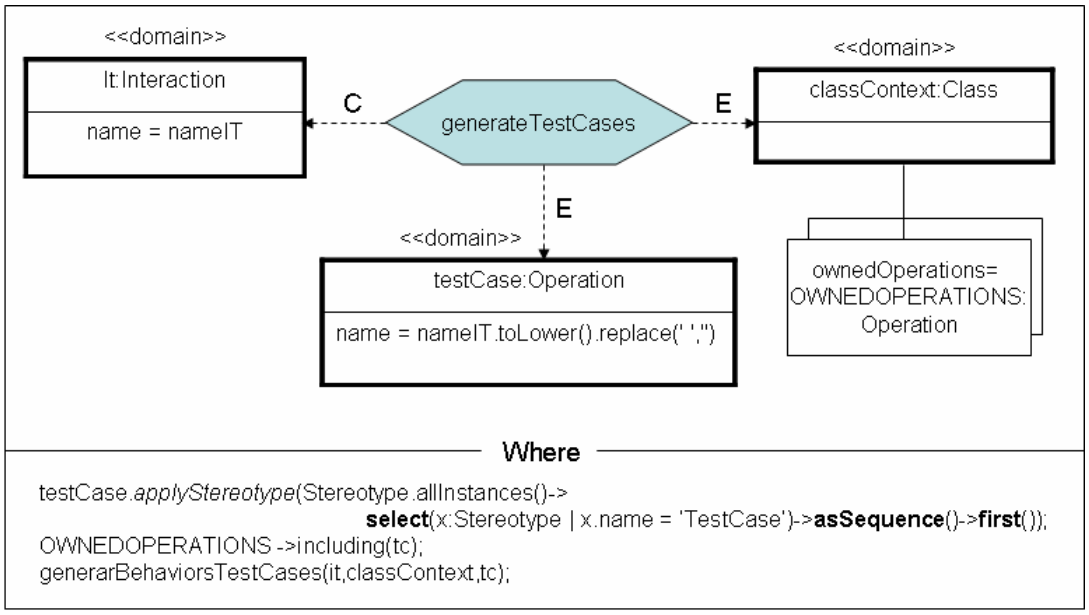

Figura 6 - Transformación para generar los diferentes "TestCase"

En el campo "where" de esta relación se hace uso de otra relación, *generarBehaviorsTestCases*, que se encarga, haciendo uso a su vez de otras relaciones QVT, de generar los diferentes elementos que modelan el comportamiento de caso de prueba (Tabla 2 y Tabla 3).

4. Ejemplo de aplicación de pruebas dirigidas por modelos

En esta sección se presenta un ejemplo de la metodología definida en la sección 3 de este artículo. El ejemplo modela el escenario de ejecución principal de la funcionalidad de autenticación de un usuario en el sistema. En un entorno real aparecerían múltiples funcionalidades con múltiples escenario, pero para este ejemplo nos centraremos en un escenario concreto.

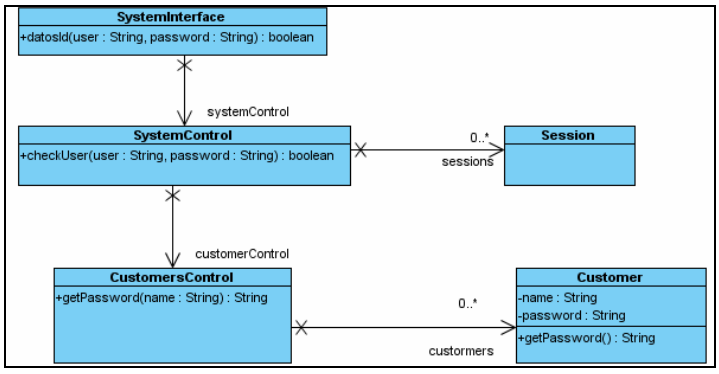

Figura 7. Diagrama de clases del sistema

La Figura 7 muestra el diagrama de clases con la arquitectura que implementa esta funcionalidad y la Figura 8 representa el diagrama de secuencia del comportamiento del sistema durante el escenario de ejecución principal. Las pre y postcondiciones que extienden el diagrama para este ejemplo están definidas en OCL y añadidas al diagrama de manera visual mediante una nota.

Obsérvese que, el contexto de la restricción OCL que se ha añadido al diagrama está centrado en un elemento del propio diagrama, de manera que esta restricción solo es aplicable en este escenario de ejecución.

Aplicando las transformaciones definidas en la Tabla 1 a estos dos elementos obtenemos la arquitectura de pruebas modelada en la

Figura 9. Obsérvese que las diferentes clases están esteriotipadas según el metamodelo de pruebas de U2TP y que diversos elementos se nombran de manera que hacen referencia a lo que se está probando, tanto la funcionalidad, como su escenario de ejecución. También se pueden observar las diferentes relaciones entre el sistema de pruebas y el sistema.

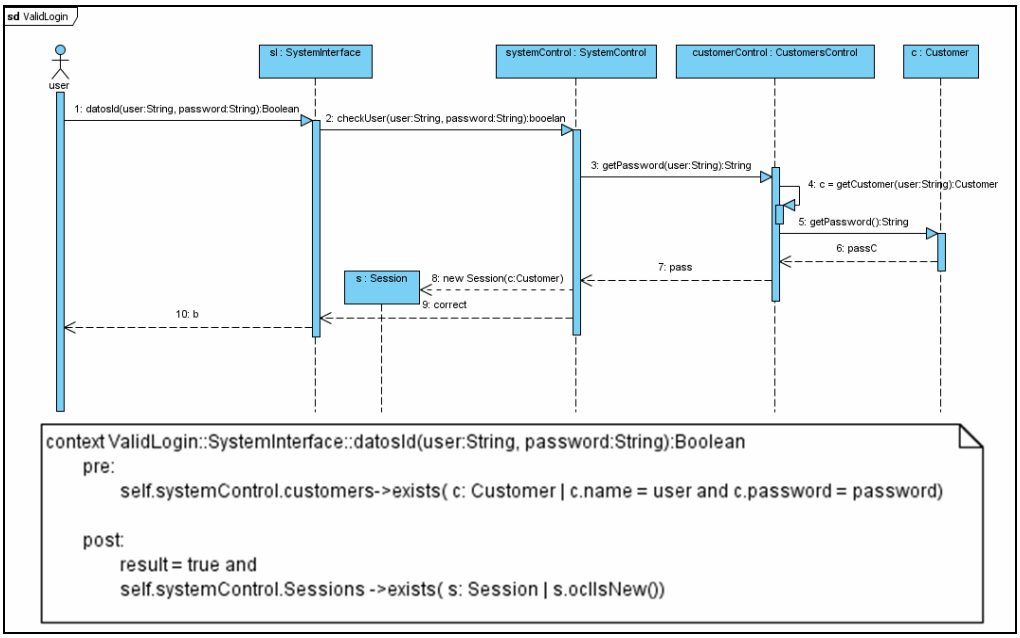

Figura 8. Diagrama de secuencia del sistema con pre y postcondiciones en OCL para las pruebas

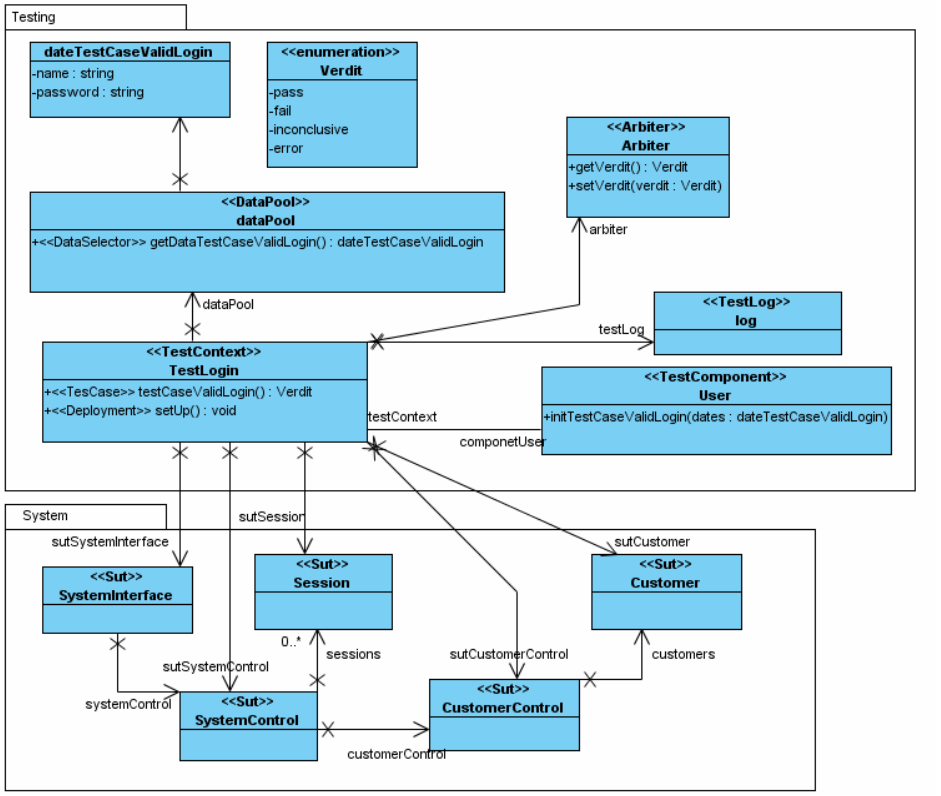

Figura 9. Modelo de clases del sistema de pruebas

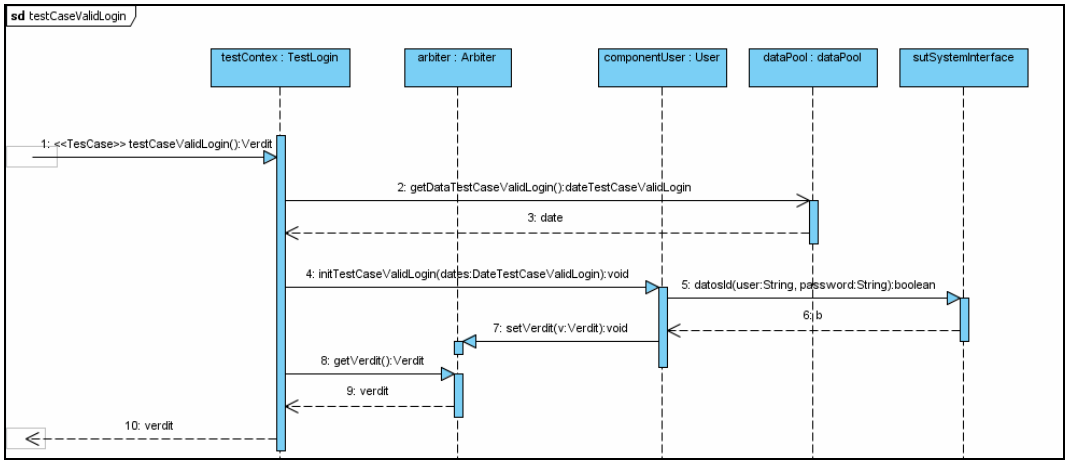

Figura 10. Diagrama de secuencia del caso de prueba

Una vez obtenida la arquitectura de pruebas, interesa modelar el comportamiento para el caso de prueba de la funcionalidad “Autenticación de un usuario al sistema”. Aplicando las transformaciones definidas en las Tabla 2 y de la

se obtienen los diferentes comportamientos de los casos de prueba, en este caso solamente para el caso de prueba “testCaseValidLogin()”. El diagrama mostrado en la

Figura 10 muestra este comportamiento. De esta forma se genera para el escenario principal de la funcionalidad de autenticación, el caso de prueba en forma automática.

5. Conclusiones y trabajo a futuro

El desarrollo de esta investigación se centra principalmente en la generación automática de pruebas siguiendo un enfoque MDA. La idea central es que los modelos de diseño del sistema como el diagrama de secuencia y el diagrama de clases, que corresponden al PIM se transforman mediante QVT en un modelo de pruebas. Este modelo de pruebas puede ser refinado luego según la plataforma del sistema, mediante transformaciones QVT, en un modelo *JUnit* por ejemplo.

Para lograr una automatización completa del proceso, se requiere contar con una especificación del oráculo de las pruebas, del cual se puedan derivar los datos de prueba en forma automática. Para esto se ha definido una extensión del metamodelo de UML donde se expresan las pre y pos condiciones de cada diagrama de secuencia mediante OCL. Actualmente se han realizado las transformaciones QVT desde el modelo del sistema al modelo de prueba. En una segunda etapa se realizarán las transformaciones desde el modelo de pruebas a un modelo dependiente de la plataforma (por ejemplo, JUnit).

Dentro del trabajo a futuro se encuentra llevar los resultados de esta investigación al ámbito de las pruebas en líneas de producto de software. La importancia de la trazabilidad entre los distintos modelos que definen la línea hace pensar que un enfoque combinado entre MDA y líneas de producto es factible. La metodología propuesta en este artículo puede ser extendida para tratar la variabilidad a nivel de los modelos de diseño del sistema y que esta variabilidad sea transformada a los modelos de prueba. De esta forma se obtendrán los casos de prueba de cada producto cuando la variabilidad sea resuelta.

Agradecimientos

Este trabajo ha sido parcialmente soportado por el proyecto PRALIN (Junta de Comunidades de Castilla-La Mancha, PAC08-0121-1374)

Referencias

[1] Schmidt, D., "Model-Driven Engineering", *IEEE Computer*, vol. 39 nº 2, pp. 25-31, 2006.

[2] Miller, J. and J. Mukerji, *MDA Guide Version 1.0. 1*, Object Management Group, 2003.

[3] Moreno N., R.J., Romero R., Vallecillo A., *Desarrollo de Software dirigido por modelos, in Fábricas de Software: experiencias, tecnologías y organización*, Ra-Ma, 2007.

[4] OMG, *Meta Object Facility (MOF) 2.0 Query/View/Transformation*, v1.0, OMG, 2008.

[5] OMG, *UML testing profile Version 1.0,* OMG, 2005.

[6] Baresi, L. and M. Young, *Test oracles*, Dept. of Computer and Information Science, Univ. of Oregon, 2001.

[7] Bertolino, A. "Software Testing Research: Achievements, Challenges, Dreams" in *International Conference on Software Engineering. 2007*.

[8] Baker, P., et al., *Model-Driven Testing: Using the UML Testing Profile*, Springer, 2007.

[9] Dai, Z., "Model-Driven Testing with UML 2.0" En *Second European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations. Canterbury, England*, 2004.

Reseña sobre el taller de Pruebas en Ingeniería del Software 2007 (PRIS)

Claudio de la Riva
Departamento de Informática
Universidad de Oviedo
claudio@uniovi.es

El III Taller sobre Pruebas en Ingeniería del Software (PRIS 2008: http://in2test.lsi.uniovi.es/pris2008/) se celebró en Gijón el 7 de Octubre de 2008, en el marco de las XIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2008). El taller se organizó como parte de las actividades de la Red para la Promoción y Mejora de las Pruebas en Ingeniería del Software (RePRIS: http://in2test.lsi.uniovi.es/repris/) financiada por el Plan Nacional de I+D+I del Ministerio de Educación y Ciencia y cofinanciada con fondos FEDER (acciones especiales TIN2005-24792-E y TIN2007-30391-E).

Este taller se configura como un foro de discusión de las actividades de I+D y de formación en relación con la prueba del software. En esta tercera edición se contó con 28 asistentes, tanto de España como de Iberoamérica. Se presentaron un total de 7 contribuciones seleccionadas tras haber sido sometidas a un proceso de revisión por pares, de las cuales dos procedieron de Iberoamérica y una como fruto de colaboración entre la Universidad y la Empresa. Así mismo, durante el taller se impartió el tutorial industrial "TMapNext – Pruebas impulsadas por los objetivos de negocio" por E. van Driel de Sogeti España.

Las contribuciones presentaron temáticas variadas. En un primer bloque se presentaron trabajos y estudios de orientación fundamentalmente investigadora. Tres de ellos están relacionados con las pruebas de composiciones de servicios web especificados con BPEL y pruebas basadas en mutación: "Framework para la generación dinámica de invariantes en composiciones de servicios web con WS-BPEL" por A. García Domínguez, M. Palomo Duarte e I. Medina Bulo, "Operadores de mutación para WS-BPEL 2.0" por A. Estero Botaro, F. Palomo Lozano e I. Medina Bulo y "Generación de mutantes con algoritmos genéticos" por J.J. Domínguez Jiménez, A. Estero Botaro e I. Medina Bulo, todos ellos de la Universidad de Cádiz. Un cuarto trabajo presenta un estudio empírico sobre la aplicación práctica de las pruebas del software en el entorno industrial: "Factores

que afectan negativamente a la aplicación práctica de las pruebas de software" por L. Fernández-Sanz de la Universidad de Alcalá, P. Lara y M.T. Villalba de la Universidad Europea de Madrid y T. Vos del Instituto Tecnológico de Informática del la Universidad Politécnica de Valencia. El último trabajo presenta una propuesta de pruebas basadas en modelos: "Propuesta para pruebas dirigidas por modelos usando el perfil de pruebas de UML 2.0" por B. Pérez Lamancha de la Universidad de la República de Uruguay y P. Reales Mateo, I. García-Rodríguez de Guzmán y M. Polo Usaola de la Universidad de Castilla-La Mancha.

En el segundo bloque de contribuciones se presentaron trabajos orientados hacia la evaluación de herramientas: "Propuesta de Modelo para Especificar la Calidad de Herramientas de Prueba" por M. Pérez, E. Méndez, K. Domínguez y L. E. Mendoza de la Universidad Simón Bolivar (Venezuela) y procesos de pruebas: "TestPAI: Un área de proceso de pruebas integrada con CMMI" por A. Sanz, J. Saldaña y J. García de la Universidad Carlos III y D. Gaitero de Atos-Origin, S.A.

Finalmente, agradecer al resto del comité organizador del taller, a los organizadores de las conferencias que albergaron este taller, a los participantes y miembros de la red RePRIS, al MEC como entidad financiadora, y en especial al editor de REICIS por el soporte e interés mostrado en este taller. Todos ellos contribuyen tanto a la consolidación de diversos grupos de investigación en pruebas como a la aportación de una visión práctica que pueda mejorar finalmente la formación y los procesos de pruebas realizados en la práctica industrial.

REICIS

ACTUALIDAD INVITADA

Apoyo del Ministerio de Industria, Turismo y Comercio (MITYC) a la modernización de PYMES del sector TIC

Carlos Fernández Gallo
Jefe de Área de Informática, Subdirección General para la Economía Digital
Ministerio de Industria, Turismo y Comercio
C/Capitán Haya, 41. 28071-Madrid
CFERNANDEZG@mityc.es

Antecedentes

En el año 2006, dentro del Plan Avanza 2006-2010, se puso en marcha por parte de la Secretaria de Estado de Telecomunicaciones y para la Sociedad de la Información, una línea de ayudas a las PYMES del sector TIC a fin de que las mismas pudiesen obtener un conjunto de certificaciones relacionadas con la calidad del software.

Las ayudas consistían en la concesión de subvenciones a fondo perdido y prestamos blandos, al cero por ciento de interés y a devolver en 7 años incluidos 2 de carencia, para las PYMES que se propusiesen como objetivo la obtención de alguna de las certificaciones de calidad siguientes:

- Capability Maturity Model Integration, CMMI ® niveles 2 a 5.
- ISO/IEC 15504, Software Process Improvement and Capability dEtermination, SPICE niveles 2 a 5.
- ISO 9001:2000, Sistemas de Gestión de la Calidad.

El importe de las ayudas por tipo de certificación obtener eran las siguientes:

- CMMI o SPICE nivel 2, 20.000 €, aumentando a 25.000 € para el nivel 3, 30.000 € para el nivel 4 y finalmente 35.000 € para el nivel 5.
- ISO 9001:2000, 6.000 €.

La motivación de la puesta en marcha de esta línea de ayudas puede apoyarse en diversas argumentaciones aunque, sin ser exhaustivo, algunas de las más representativas serian:

- La existencia de un sector TIC con un nivel de desarrollo suficientemente elevado que aconsejaba dar un salto diferencial, perfeccionando en las PYMES unas metodologías de calidad de acuerdo con estándares internacionales de reconocido prestigio.
- El interés suscitado, tanto en España como en países de nuestro entorno, como posibles receptores de factorías de software *nearshore*, frente a las más tradicionales factorías *offshore*, lo cual aconsejaba disponer en las PYMES españolas de certificaciones relativas a la calidad del software, como un elemento fundamental de interlocución.
- El papel creciente que esta tomando el software con una mayor presencia en todo tipo de productos, procesos y servicios, algunos de ellos en sistemas críticos, lo cual obliga a extremar cada vez más los criterios de calidad de los desarrollos efectuados, buscando trabajar siempre con metodologías industriales.

La línea de ayudas permitía obtener la certificación en una o en dos anualidades, en función de la mayor o menor dificultad de obtención de la misma, del grado de experiencia de las empresas en la implantación de normas dentro de sus organizaciones y del tipo de implantación deseada, mas rápida o mas lenta en función de la mayor o menor disponibilidad de RRHH propios a destinar al proceso de certificación. Adicionalmente, se establecía que los proyectos se deberían gestionar de forma colaborativa entre una Asociación, Agrupación o entidad equivalente que prestase servicios de apoyo a las PYMES y estas. Esta Asociación lógicamente debía tener un papel activo antes y durante el desarrollo del proyecto.

Antes del inicio del proyecto, realizando un diagnostico previo de la situación del proceso de software de las PYMES, una definición de los procesos de mejora a llevar a cabo y la determinación del tipo de certificación a obtener por las mismas.

Durante el desarrollo del proyecto, ayudando a la implantación de los procesos de calidad internos previos a la obtención de la certificación y en definitiva ayudando a la obtención de la misma.

Es evidente la importancia de la Asociación o equivalente como elemento de apoyo a las PYMES, canalizando y controlando adecuadamente las tareas a realizar, lógicamente en colaboración estrecha con la entidad consultora y certificadora. Por estas funciones de apoyo la Asociación recibía un 15% de la ayuda total recibida por las PYMES participantes en el proyecto.

Situación actual

La experiencia durante los dos primeros años de vigencia 2006 y 2007 fue satisfactoria, permitiendo a un colectivo grande de PYMES abordar nuevas metodologías de trabajo de acuerdo con los estándares internacionales antes descritos.

Por ello, en el año 2008 se decidió ampliar el ámbito de actuación de la línea de ayudas, incorporando dos nuevas tipologías de certificaciones vinculadas con la calidad del software. Se seguía manteniendo la posibilidad de realizar el proyecto anual o bianualmente, con una Agrupación y/o Asociación que actuase como coordinador del mismo. Las ayudas antes indicadas relativas a la calidad del software y las de estas dos nuevas certificaciones se concedieron únicamente vía subvención.

Estas certificaciones son:

- UNE ISO/IEC 20000-1:2005, Gestión del Servicio TI.
- ISO/IEC 27001:2005, Gestión de la Seguridad de la Información.

El importe de las ayudas para estas dos nuevas certificaciones es:

- UNE ISO/IEC 20000-1:2005, 20.000 €.
- ISO/IEC 27001:2005, 19.000 €.

Con estas dos nuevas certificaciones, la primera de ellas muy orientada a las PYMES que prestan servicios TI y la segunda de gran importancia en todas las PYMES TIC, con independencia de si su actividad básica es desarrollar productos de software o prestar servicios TI, se completa el ámbito de certificaciones, a apoyar con ayudas publicas, que hoy en día cada vez están tomando mayor peso especifico dentro del sector.

El balance final de la convocatoria de ayudas del año 2008 demuestra una cierta tendencia de consolidación en el interés de las PYMES en abordar alguna de las certificaciones anteriores, como un elemento, no ya de prestigio sino de utilidad real a la hora de abordar sus procesos de negocio, siendo cada vez mas conscientes que la rentabilidad al implantar una mejora de sus procesos productivos siempre es a medio y largo plazo. Prueba de lo anterior es el número de PYMES apoyadas:

- En Calidad del Software 102 PYMES (62 de ellas para obtener CMMI o SPICE niveles 2 y 3 y 40 para obtener ISO 9001).
- En Gestión del Servicio TI, 49 PYMES.
- En Gestión de la Seguridad de la Información, 118 PYMES.

Hay que mencionar como aspecto interesante la localización geográfica de estas PYMES apoyadas, con gran abundancia de ellas situadas fuera de Madrid y Cataluña, las dos zonas geográficas con mayor implantación sectorial, lo cual demuestra un interés creciente en las mismas por mejorar su situación competitiva. Finalmente también mencionar el incremento de empresas TIC certificadas según CMMI, desde la puesta en marcha de esta línea de apoyo en el 2006, lo cual sitúa a España en estos momentos como segundo país de la UE en cuanto a empresas certificadas, solo superada por de Francia, situación muy diferente a la existente en el año 2005.

Referencias

Más información en www.mityc.es

Perfil profesional

Carlos Fernandez Gallo es Jefe de Área de Informática de la Subdirección General para la Economía Digital del Ministerio de Industria, Turismo y Comercio.

Su actividad profesional en este campo se ha centrado en el análisis y evaluación de proyectos de I+D y los relativos a obtención de certificaciones, presentados por las empresas TIC a efectos de obtención de ayudas publicas.