Revista
Española de

Innovación, Calidad e Ingeniería del Software

REICIS

Volumen 5, Número 2 (especial XI JICS), septiembre, 2009

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

Publicado por la Asociación de Técnicos de Informática (ATI), Via Laietana, 46, 08003 Barcelona.

Secretaría de dirección: ATI Madrid, C/Padilla 66, 3º dcha., 28006 Madrid

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

Contenidos

REICIS

Editorial

REICIS

El Grupo de Calidad del Software de ATI (www.ati.es/gtcalidadsoft) sigue consolidado su posición como principal promotor de la disciplina de ingeniería y calidad del software con la undécima edición de las Jornadas sobre Innovación y Calidad del Software (las tradicionales JICS). Con su actividad desde 1997 (véase en su web la sección de actividades que contiene todo su recorrido histórico), el grupo de Calidad del Software de ATI es la entidad más veterana en este ámbito en España. Estas XI JICS siguen potenciando la presencia iberoamericana en este foro de promoción de la cultura de la calidad del software y de la innovación en el desarrollo de sistemas y aplicaciones en la ya segunda edición de la Conferencia Iberoamericana de Calidad del Software (CICS). Por otra parte, las XI JICS incorporan la presencia de la ponencia de una destacada referencia en el ámbito europeo de la ingeniería de software como es Margaret Ross, SQM, BCS y editora de la conocida revista Software Quality Journal.

Como es habitual y gracias a la colaboración del comité de programa, se sigue cuidando la calidad de los trabajos presentados, eminentemente prácticos pero rigurosos, aceptados entre los remitidos en la convocatoria de contribuciones: las ponencias aceptadas (con una tasa de aceptación del 41,7%) han sido sometidos a un completo proceso de revisión. Por supuesto, no cabe olvidar el apoyo de instituciones como la Universidad de Alcalá que ha contribuido a su organización no sólo aportando la sede sino un apoyo financiero a través de su Vicerrectorado de Investigación. También se debe destacar el patrocinio de DEISER, no sólo aportando recursos sino también una interesante presentación de experiencias innovadoras para el mundo SAP con UML. No podemos olvidar tampoco la presentación de novedades como el nuevo estándar de pruebas de software ISO 29119 . En definitiva, el evento más completo con toda la información disponible en la página del grupo de Calidad del Software (www.ati.es/gtcalidadsoft) acorde a su trayectoria pionera en España, está proporcionando, a través de la Asociación de Técnicos de Informática, el apoyo para la productividad y la calidad en los proyectos de software.

Luis Fernández Sanz
Juan J. Cuadrado-Gallego
Editores

Presentación

REICIS

En este número especial de septiembre de 2009 de REICIS, por segunda vez en la historia de nuestra revista, esta publicación se convierte en el vehículo de difusión del evento decano en España en el ámbito de la ingeniería y la calidad del software: las Jornadas de Innovación y Calidad del Software (JICS) que alcanzan así su undécima edición desde su inicio en 1998. De nuevo, el Grupo de Calidad del Software de ATI (www.ati.es/gtcalidadsoft) ha cuidado que los trabajos aceptados hayan sido sometidos a un completo proceso de revisión por el comité de programa. En esta edición se han incluido dos tipos de trabajos. Por una parte, se incluyen los trabajos clásicos de investigación e innovación con extensión máxima de 10 paginas centrados en avances soportados por las prácticas comunes de justificación y método científico. Por otra, las experiencias de aplicación práctica en entornos industriales de mejoras y métodos de calidad del software: estos trabajos cuentan con un formato más corto con un máximo de 6 páginas y una orientación eminentemente práctica. Entre los incluidos en el número, los trabajos correspondientes a esta categoría son los de la siguiente lista:

- "Visión Innovadora de la Calidad del Producto Software"
- "El análisis de anomalías detectadas en las pruebas de software: una vía para mejorar el ciclo de vida"
- "Experiencias de una PYME en la Mejora de Procesos de Pruebas"
- "Procedimiento para pruebas de intrusión en Aplicaciones Web"

Este número especial constituye en definitiva la publicación de las actas de las XI JICS y, por ello, cuenta con un tamaño mayor del habitual. Desde aquí agradecemos la labor del comité de programa compuesto por la siguiente lista de expertos:

- Antonia Mas (Universitat de les Illes Balears)
- Miren Idoia Alarcón Rodríguez (Universidad Autónoma de Madrid)
- Javier Tuya (Universidad de Oviedo)
- María Moreno (Universidad de Salamanca)
- José Antonio Calvo-Manzano (Universidad Politécnica de Madrid)
- Esperança Amengual (Universitat de les Illes Balears)
- José Ramón Hilera (Universidad de Alcalá)
- María José Escalona (Universidad de Sevilla)
- Tanja Vos (Universidad Politécnica de Valencia)
- Carmes Pages (Universidad de Alcalá)
- José Luis Baéz (ESPELSA)
- Pilar Romay (Fundación Giner de los Ríos)

Luis Fernández Sanz

Analizando el apoyo de marcos SPI a las características de calidad del producto ISO 25010

César Pardo, Francisco J. Pino
Grupo IDIS, Facultad de Ing. Electrónica y Telecomunicaciones
Universidad del Cauca
{cpardo, fjpino}@unicauca.edu.co
Félix García, Mario Piattini
Departamento de Tecnologías y Sistemas de Información, Escuela Superior de Informática
Universidad de Castilla-La Mancha.
{Felix.Garcia, Mario.Piattini }@uclm.es

Resumen

Actualmente, por las condiciones de tiempo, costo y recursos que involucran la implementación de los marcos de SPI en las organizaciones desarrolladoras de software, es necesario identificar y establecer las características basadas en la mejora no sólo de los procesos sino también desde el contexto de la calidad de producto que motiven a las empresas a adoptar el enfoque más adecuado a sus necesidades. Con el fin de presentar una forma de seleccionar los marcos para la mejora de los procesos de las organizaciones basada en los requisitos de calidad de producto de una organización, en este trabajo se analiza los estándares ISO 90003 y CMMI-DEV v1.2 con el objetivo de identificar en qué medida las entidades de procesos descritas en estos marcos, apoyan la consecución de las características y subcaracterísticas de calidad del producto software descritas en la norma ISO 25010 de SQuaRE.

Palabras clave: ISO 25010, CMMI, ISO 90003, comparación, armonización e integración de marcos para la mejora de procesos.

Analyzing support of SPI frameworks to product quality characteristics ISO25010

Abstract

Currently, given the time, cost and resources conditions involved in the implementation of SPI frameworks in software development organizations, it is necessary to identify and to establish the quality characteristics which can be improved not only from the perspective of the processes but also from the perspective of the product. This can motivate companies to adopt the process improvement approach which is the best-suited to their product quality requirements. In order to support the selection of process improvement frameworks suited to product quality needs, this paper examines the standards ISO 90003 and CMMI-DEV v1.2 with the aim to identify up to what extent they cover the characteristics and subcharacteristics of software product quality as described in ISO 25010 of SQuaRE.

Key words: ISO 25010, CMMI, ISO 90003, comparison, standards harmonization and integration.

Pardo C., Pino, F.J., García, F. y Piattini, M.,"Analizando el apoyo de marcos SPI a las características de calidad del producto ISO 25010", REICIS, vol. 5, no.2, 2009, pp.6-16. Recibido: 22-6-2009; revisado: 6-7-2009; aceptado: 31-7-2009

1. Introducción

Encontrar un marco[1] de calidad lo suficientemente completo y que satisfaga las necesidades de todas las organizaciones que desarrollan software es imposible. Ashrafi en [1] plantea un ejemplo muy sencillo para comprender mejor esto; "una organización que produce software de misión crítica, considera la fiabilidad la característica más importante, mientras que la portabilidad puede ser la más prioritaria en una organización que produce software para una gran variedad de plataformas". Por tanto, "lo que se considera un objetivo importante para una organización puede no ser importante para otra". Es por esto que se aborda la pregunta que inquieta tanto a las organizaciones que se encuentran en la selección del marco que solucione sus necesidades empresariales y/o requisitos de calidad: ¿Alguno de los marcos de SPI existentes satisface mis requisitos de calidad?, ¿Es suficiente un marco de SPI para satisfacer dichos requisitos? Alrededor de las anteriores preguntas, en [2], [3] y [4] se han realizado algunos estudios empíricos que demuestran qué marcos de calidad definidos por el SEI como CMMI o ISO con ISO 90003 mejoran la calidad de los productos software con base a un conjunto de características de calidad. Asimismo, la heterogeneidad de los marcos habilita la oportunidad de trabajar con ellos de forma complementaria [5].

Un aspecto a tratar en la ingeniería de Software es cómo los procesos apoyan la calidad de los productos. Es decir, cómo los marcos definidos para soportar la mejora de los procesos de una organización, apoyan el cumplimiento de los objetivos de calidad del producto software a desarrollar. La identificación de la relación entre modelo, proceso, producto y característica de calidad, probablemente facilitaría la mejora de procesos con la selección de marcos adecuados a las necesidades de una organización.

No obstante, aun se sigue sin resolver la forma de cómo las organizaciones puedan seleccionar un marco de SPI considerando/teniendo en cuenta las características de calidad del producto software que desarrollan. Es por esto que para la elaboración de este trabajo nos preguntamos: ¿Podrían las características y subcaracterísticas de calidad de producto de

[1] En adelante para unificar los diferentes términos usados por las diferentes organizaciones y autores para referirse a los modelos, usaremos el término genérico "marco" para hacer referencia a: modelos y estándares.

la norma ISO 25010, ser utilizadas como base para examinar y elegir el marco de SPI más adecuado a los requisitos de calidad de una organización? En un primer esfuerzo por identificar en que medida se tratan las características de calidad en los marcos para la mejora de procesos de software ISO 90003 [6] y CMMI-DEV V1.2 [7], en este trabajo se lleva a cabo un análisis de las subcaracterísticas de calidad según la norma ISO 25010 [8] y las cláusulas y objetivos específicos (SG) respectivamente. Como resultado de este análisis, se obtiene un árbol de decisión para la selección del marco de SPI más adecuado según los requisitos o necesidades de calidad de producto software de una organización.

Además de la presente introducción, el artículo presenta: en la sección 2 los pasos realizados en el análisis de las subcaracterísticas de calidad de ISO 25010 y los marcos de SPI. La sección 3 muestra el análisis de ISO 90003 y CMMI con relación a las subcaracterísticas de calidad. La sección 4 presenta un árbol de decisiones para la selección de marcos de SPI. Y la sección 5 muestra las conclusiones y trabajos futuros.

2. Pasos para el análisis de subcaracterísticas de calidad

La norma ISO 25010 provee un marco de referencia para medir la calidad del producto software y describe 8 características y 38 subcaracterísticas de calidad de producto software. Usamos este marco como referente para analizar cómo las entidades de proceso descritas en ISO 90003 y CMMI, pueden brindar soporte a las subcaracterísticas de calidad definidas en él.

Para realizar el análisis, fue importante tener en cuenta que las subcaracterísticas de calidad del producto están definidas desde el punto de vista del usuario final, y los marcos de SPI definidos desde el punto de vista de los desarrolladores. Es decir, que los marcos de SPI están más orientados a recomendar buenas prácticas de los procesos y actividades realizadas por los desarrolladores de software, y los marcos de calidad como ISO 25010 a las características de calidad que están orientadas al producto.

Para llevar a cabo el análisis de cómo las subcaracterísticas de ISO 25010 están apoyados con las cláusulas de la norma ISO 90003 y los Objetivos Específicos (SG) de CMMI-DEV, se llevaron a cabo los siguientes pasos:

- Analizar si las cláusulas de ISO 90003 y los SG de CMMI tienen relación o apoyan a una subcaracterística de calidad.

- Documentación de las diferencias y similitudes de las cláusulas y SG respecto a las definiciones de las subcaracterísticas de calidad.
- Hallar el apoyo de acuerdo a la cantidad de cláusulas y SG encontrados por cada subcaracterística.

3. Análisis de ISO 90003 y CMMI con ISO 25010

Para realizar el análisis, decidimos usar ISO 90003 y CMMI-DEV por ser marcos para SPI muy usados, difundidos y estar orientados al desarrollo de software.

El análisis de las cláusulas ISO 90003 y SG de CMMI se enfocó en la identificación de sinónimos o palabras que refirieran información relacionada con la definición de las subcaracterísticas de calidad. Asimismo, los comentarios generados y que facilitaban la comprensión del análisis realizado, también fueron documentados.

En la tabla 1, se presenta la plantilla donde se recolectó la información del análisis realizado sobre las subcaracterísticas. Por motivos de espacio, sólo se presenta la información de la subcaracterística de calidad de *exactitud*.

Subcaracterística	Cláusulas ISO 90003	Áreas de proceso claves CMMI-DEV	Comentarios
Exactitud. El grado por el cual el producto software provee el correcto o resultado especificado y con el grado de precisión necesario.	7.3.4 Revisión del diseño y desarrollo ... 7.3.5. Verificación del diseño y desarrollo … 7.3.6 Validación del diseño y desarrollo … 7.3.6.1 Validación y 7.3.6.2 Pruebas.	Objetivos específicos de las áreas de proceso: Integración del Producto (3) Validación (2) Verificación (3)	En ISO 9003 se analizaron las cláusulas que relacionan recomendaciones para verificar o probar el producto software en el cumplimiento del resultado especificado. En CMMI-DEV la integración de los diferentes componentes del producto y su respectiva verificación y validación, conducen al correcto resultado especificado del producto software.

Tabla 1. Análisis de la subcaracterística de calidad Exactitud con IS0 90003 y CMMI

En la tabla 1, se puede apreciar que para la *exactitud*, en ISO se encontraron 4 cláusulas relacionadas. De las cláusulas encontradas, 3 de ellas refieren información de forma implícita y sólo una se subdivide en cláusulas más explícitas. Este es el caso de la cláusula 7.3.6 referente a la validación del diseño y desarrollo, la cual relaciona cláusulas más detalladas referentes a la validación y pruebas. Por otra parte, en CMMI se encontraron 8 SG relacionados. Los SG proveen información implícita. Por razones de espacio en la Tabla 1 sólo se presenta la cantidad de SG encontrados.

En general, al realizar el análisis se tuvo especial cuidado con la sintaxis de cada marco y su relación con respecto a las subcaracterísticas. Por ejemplo, al igual que en [1], en la norma ISO 90003 la facilidad de uso se analizó a través de la búsqueda de recomendaciones que apoyaran la formación o capacitación en el uso del producto. Asimismo, el *comportamiento en el tiempo*, es adaptado con relación al comportamiento del producto software en condiciones especiales y similares al ambiente final de ejecución antes de ofrecer el producto para que sea aceptado por el cliente.

En total fueron analizadas 30 subcaracterísticas de calidad. Puesto que nos interesa analizar las subcaracterísticas desde un punto de vista técnico y no desde la perspectiva genérica del cumplimiento de las características, excluimos las 8 subcaracterísticas referentes al grado de conformidad de cada característica.

En la figura 1, se muestra el apoyo de las cláusulas y los SG sobre las características de calidad. Los valores presentados, son hallados a partir de la suma de relaciones (cláusulas o SG encontrados en el análisis de subcaracterísticas) por característica. Es posible observar que la norma ISO 90003 ofrece mayor apoyo sobre la mayoría de características de calidad, siendo menor sólo en las características de *adecuación funcional* y *fiabilidad,* con 8 y 6 cláusulas en comparación a los 10 y 8 SG definidos en CMMI. Entre las características con mayor apoyo por parte de ISO se encuentran la *eficiencia de rendimiento* (5), *operabilidad* (24), *compatibilidad* (4), *mantenibilidad* (22) y *transferabilidad* (7).

En la figura 2 se presenta un análisis más detallado con relación a las subcaracterísticas de calidad que componen las características de la figura 1. Los valores presentados, representan el número de relaciones encontradas por cada subcaracterística. Este análisis revela que los marcos de SPI no cubren la totalidad de las subcaracterísticas de calidad. De ese modo se puede observar un bajo apoyo en las subcaracterísticas relacionadas con la s*eguridad*, por ejemplo, ISO 90003 y CMMI cubren sólo la *confidencialidad* (3) y la *integridad* (3) respectivamente.

También se puede observar un conjunto de subcaracterísticas que no son cubiertas por los dos marcos, es el caso de la *adaptabilidad, estabilidad de modificación, reemplazabilidad, autenticidad, responsabilidad, no repudio* o *no rechazo* y *atractividad.*

Analizando las características que sólo son abordadas por un marco, es posible observar que ISO apoya 9 subcaracterísticas de calidad, ellas son: *portabilidad* (3), *cambiabilidad* (4), *reusabilidad* (1), *interoperabilidad* (4), *confidencialidad* (3), *accesibilidad técnica* (1), *utilidad* (4), *utilización de recursos* (2) y *recuperabilidad* (1). Por su parte, CMMI cubre sólo 3 subcaracterísticas, la *modularidad* (6), *coexistencia* (3) y la *integridad* (3).

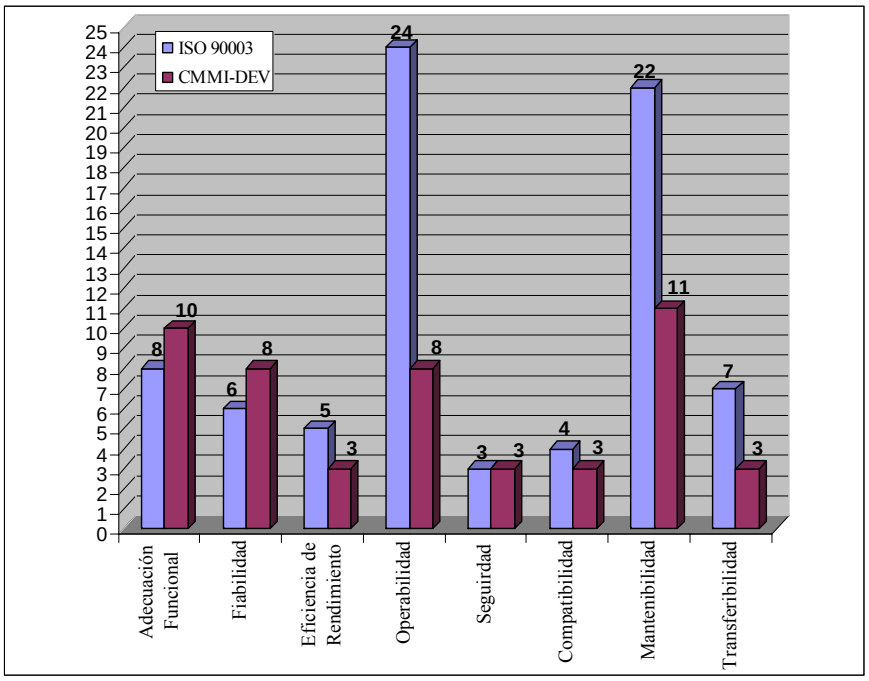

Figura 1. Análisis de ISO 90003 y CMI-DEV V1.2 con las características de ISO 25010

Las subcaracterísticas referentes a la *instalabilidad, facilidad de prueba, analizabilidad, facilidad de uso, facilidad de aprendizaje, pertinencia reconocible, comportamiento en el tiempo, tolerancia a fallos, disponibilidad, exactitud* e *idoneidad*, son cubiertas por los dos marcos. Sin embargo, es posible notar diferentes grados de especialización o detalle. Por ejemplo, CMMI ofrece mayor apoyo a las subcaracterísticas de *disponibilidad* y *exactitud*, con una diferencia de 3 y 5 relaciones más con respecto a ISO. Por su parte, ISO ofrece mayor apoyo sobre la *instalabilidad, facilidad de prueba, facilidad de uso, facilidad de aprendizaje, pertinencia reconocible* y la *idoneidad.* Siendo la *facilidad de prueba, facilidad de aprendizaje, pertinencia reconocible* y la *idoneidad* las subcaracterísticas con mayor diferencia en cuanto al número de relaciones encontradas, de 15, 9, 6 y 5 respectivamente.

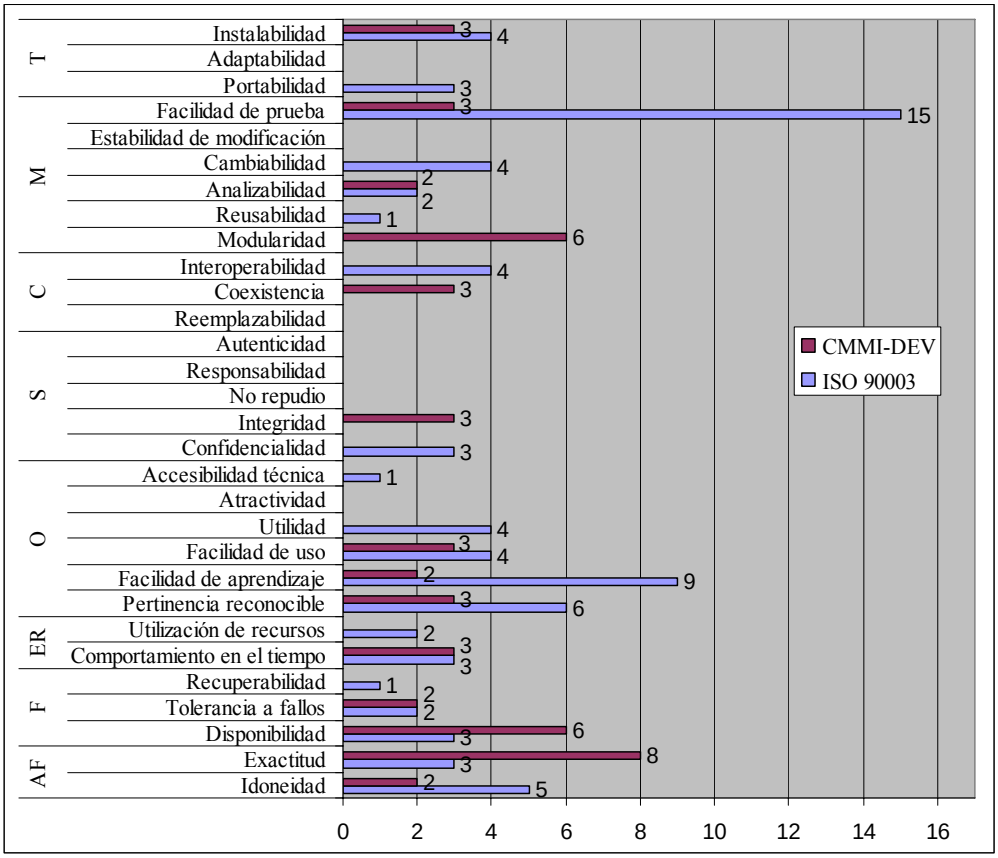

Figura 2. Análisis de ISO 90003 y CMI-DEV V1.2 con las subcaracterísticas de ISO 25010

En la tabla 2 es posible observar de forma más detallada el número de cláusulas y SG encontrados en cada marco. Al parecer, la cantidad de cláusulas en ISO es mayor que los objetivos específicos en CMMI. Esto quizá es debido a que el análisis realizado sólo tiene en cuenta los objetivos específicos de las áreas claves en CMMI, obviando las prácticas específicas asociadas a estos, mientras que en ISO 90003 por ser una norma menos extensa, se tuvieron en cuenta todas las cláusulas. Como trabajo futuro, se abordará CMMI a nivel de sus prácticas específicas.

En total, de las 30 subcaracterísticas de calidad analizadas, en ISO 90003 se encontraron 79 cláusulas y en CMMI 49 SG relacionados. ISO 90003 ofrece apoyo en 20 subcaracterísticas a diferencia de CMMI que sólo lo hace en 14.

	Subcaracterísticas de calidad ISO 25010																														
	FS		R			PE		O						S					C			M						T			
	Idoneidad	Exactitud	Disponibilidad	Tolerancia a fallos	Recuperabilidad	Comportamiento en el tiempo	Utilización de recursos	Pertinencia reconocible	Facilidad de aprendizaje	Facilidad de uso	Utilidad	Atractividad	Accesibilidad técnica	Confidencialidad	Integridad	No repudio	Responsabilidad	Autenticidad	Reemplazabilidad	Coexistencia	Interoperabilidad	Modularidad	Reusabilidad	Analizabilidad	Cambiabilidad	Estabilidad de modificación	Facilidad de prueba	Portabilidad	Adaptabilidad	Instalabilidad
ISO	5	3	3	2	1	3	2	6	9	4	4	0	1	3	0	0	0	0	0	0	4	0	1	2	4	0	15	3	0	4
CMMI	2	8	6	2	0	3	0	3	2	3	0	0	0	0	3	0	0	0	0	3	0	6	0	2	0	0	3	0	0	3

Tabla 2. Número de cláusulas en IS0 90003 y objetivos específicos en CMMI

4. Árbol de decisión para la selección de marcos de SPI

Basado en las prácticas y recomendaciones de los marcos para SPI analizados, hemos obtenido un árbol de decisión con el cual es posible facilitar la selección de marcos dependiendo de los requisitos de calidad del producto software que desarrolle una organización. La figura 3 relaciona las subcaracterísticas y los marcos de calidad analizados. Los diferentes estados de selección pueden interpretarse de la siguiente manera: (i) la combinación de los marcos, (ii) la elección entre uno de los dos marcos, (iii) sólo puede ser usado un marco y (iv), debe ser usado otro marco.

El orden en el que son presentados los marcos en la figura 3, determina su nivel de apoyo. Es decir, el marco ubicado en la primera posición de izquierda a derecha ofrece mayor apoyo que el segundo. Las subcaracterísticas que no son relacionadas con ninguno de los marcos, deben hacer uso de cualquier otra metodología o marco que supla esa necesidad o requisito.

Un ejemplo de aplicación del árbol sería el siguiente: si el *comportamiento en el tiempo* es una subcaracterística de calidad de interés en el desarrollo del producto software de una organización, tanto CMMI como ISO apoyarían esta subcaracterística. Sin embargo, CMMI provee mayor apoyo implícito que ISO. De este modo la organización podría optar por usar sólo los SG de CMMI que cubren esta subcaracterística, o integrar los dos enfoques de manera que se alcance un mayor apoyo (ver tabla 3). Por ejemplo, en un sistema de calidad ISO 9001 institucionalizado en una organización, el complemento o integración de sus prácticas con respecto a CMMI sería una buena opción para mejorar el apoyo a las necesidades de calidad de producto software que se desarrolla.

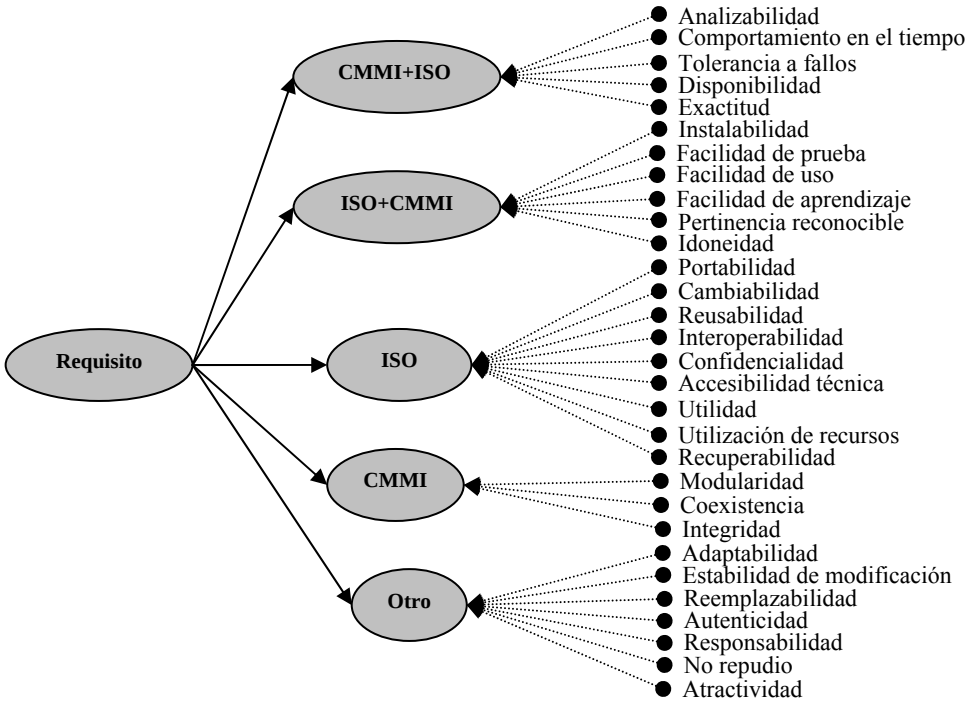

Figura 3. Árbol de decisión basado en ISO 25010 para la selección de marcos de SPI

Comportamiento del tiempo.	
Cláusulas ISO 90003	SG de CMMI-DEV
7.3.6.1 Validación. Antes de ofrecer el producto para la aceptación del cliente, conviene que la organización valide la operación del producto… 7.3.6.2 Pruebas. La validación a menudo puede realizarse mediante pruebas. … 7.6. Control de los dispositivos de seguimiento y de medición…	SG 1. La preparación para la verificación se conduce. SG 2. Las revisiones paritarias se realizan en productos del trabajo seleccionado. SG 3. Los productos del trabajo seleccionado se verifican contra sus requisitos especificados.

Tabla 3. Cláusulas y SG relacionados con la subcaracterística de calidad de Comportamiento del Tiempo

5. Conclusiones

El desarrollo de este trabajo ha permitido aplicar conceptos relacionados pero muy poco trabajados de manera integrada: la calidad del producto software y las buenas prácticas de procesos utilizados para desarrollarlo. La calidad, por ser un concepto multidimensional, posee diferentes definiciones dependiendo del contexto desde donde se examine. Es por esto que, implementar los conceptos definidos para la evaluación de la calidad de los productos software en los marcos de SPI, es una tarea que permite evaluar la selección de dichas metodologías desde otra perspectiva, no sólo desde los procesos software de una organización, sino también desde el beneficio del producto y por ende, del usuario.

En esta dirección, el trabajo aquí desarrollado, ha descrito un análisis del apoyo de marcos SPI a las características de calidad del producto ISO 25010. Con base a los resultados obtenidos, se ha elaborado un árbol de decisión que puede ser usado para la selección del marco de SPI más acorde a los requisitos de calidad del producto software de una organización.

Teniendo en cuenta que el apoyo a las características y subcaracterísticas del estándar ISO 25010 difieren dependiendo del marco a examinar, esperamos poder extender este mismo método de análisis a otros marcos que mejoren el árbol de decisión.

El análisis realizado es subjetivo y está determinado e influenciado de acuerdo a nuestras interpretaciones individuales. Adicionalmente, se llevará a cabo un estudio empírico que permita comparar y analizar las subcaracterísticas de calidad desde la opinión de expertos y/o personas involucradas en el uso de los marcos de SPI en algunas organizaciones. Esta validación permitiría comprobar su correspondencia desde un punto de vista no solo teórico, sino también empírico-práctico.

Agradecimientos

Este trabajo ha sido financiado por los proyectos: ESFINGE (TIN2006-15175-C05-05, MEC de España), Entorno colaborativo de apoyo a la mejora de procesos para la industria de software colombiana (3531-403-20708, Colciencias de Colombia) y ARMONIAS (PII2I09-0223-7948, JCCM de España).

Referencias

[1] Ashrafi, N., "The impact of software process improvement on quality: in theory and practice". *Inf. Manage*, vol. 40, nº 7, pp. 677-690, 2003.

[2] Harter, D.E., M.S. Krishnan, y S.A. Slaughter, "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development". *Manage Sci.*, vol. 46, nº 4, pp. 451-466, 2000.

[3] Krishnan, M.S. y M.I. Kellner, "Measuring Process Consistency: Implications for Reducing Software Defects". *IEEE Trans. Softw. Eng.*, vol. 25, nº 6, pp. 800-815, 1999.

[4] Krishnan, M.S., C.H. Kriebel, S. Kekre, y T. Mukhopadhyay, "An Empirical Analysis of Productivity and Quality in Software Products". *MANAGEMENT SCIENCE*, vol. 46, nº 6, pp. 745-759, 2000.

[5] Pardo, C., F. Pino, F. García, y M. Piattini. "Homogenización de marcos en ambientes de mejora de procesos multimarco". En: A. Brogi, J. Araújo, y R. Anaya. (Eds.), *XII Conferencia Iberoamericana de Ingeniería de Requisitos y Ambientes de Software*. Colombia (Medellín), 13-17 de Julio de 2009, pp. 153-166, 2009.

[6] International Organization for Standardization, *ISO/IEC 90003:2004 Software engineering - Guidelines for the application of ISO 9001:2000 to computer software*, ISO, 2004.

[7] Software Engineering Institute, *CMMI for Development V1.2 Technical Report CMU/SEI-2006-TR-008*, SEI, 2006.

[8] International Organization for Standardization, *FCD 9126-1.2 Information Technology - Software product quality - Part 1: Quality model*, ISO, 1998.

Generación automática de casos de prueba para Líneas de Producto de Software

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

Macario Polo
Grupo Alarcos, Departamento de Tecnologías y Sistemas de Información,
Universidad de Castilla-La Mancha, Ciudad Real, España
macario.polo@uclm.es

Resumen

La generación automática de casos de prueba a partir de modelos de diseño para Líneas de Producto de Software requiere definir los mecanismos para gestionar la variabilidad en las pruebas y su trazabilidad a los demás artefactos de desarrollo. En este trabajo, los casos de prueba se generan automáticamente mediante el lenguaje de transformación QVT a partir de diagramas de secuencia extendidos para representar la variabilidad en la LPS. La trazabilidad entre los distintos modelos es gestionada mediante la definición de un Perfil de UML para el Modelo de Variabilidad Ortogonal.

Palabras clave: Línea de Productos Software, Ingeniería Dirigida por Modelos, Pruebas.

Automatic test case generation for software product lines

Abstract

The automatic generation of test cases from design models in Software Product Lines requires defining testing mechanisms for managing variability and traceability to other development artifacts. In our proposal, test cases are generated using the transformation language QVT from extended sequence diagrams representing the variability in the SPL. Traceability between different models is managed by defining a UML Profile for the Orthogonal Variability Model.

Key words: Software Product Line, Model Driven Engineering, Testing

Pérez-LaMancha, B. y Polo, M.," Generación automática de casos de prueba para Líneas de Producto de Software", REICIS, vol. 5, no.2, 2009, pp.17-27. Recibido: 22-6-2009; revisado: 6-7-2009; aceptado: 31-7-2009

1. Introducción

Una línea de productos software (LPS) se define como "un conjunto de sistemas software, que comparten un conjunto común de características (*features*), las cuales satisfacen las necesidades específicas de un dominio o segmento particular de mercado, y que se

desarrollan a partir de un sistema común de activos base (*core assets*) de una manera preestablecida" [3]. Uno de los aspectos distintivos de las LPS frente al desarrollo tradicional es la importancia de la variabilidad a lo largo de todo el proceso de desarrollo: los productos de la línea comparten un conjunto de características (*commonalities*) y difieren en determinados puntos de variación (*variation points*), que representan la variabilidad entre los productos. Un aspecto central en el desarrollo de LPS es la división de los procesos de ingeniería: la Ingeniería de Dominio, responsable de desarrollar los elementos comunes al dominio y su mecanismo de variabilidad, y la Ingeniería de la Aplicación donde se desarrollan los productos concretos reutilizando los recursos creados en la Ingeniería del Dominio.

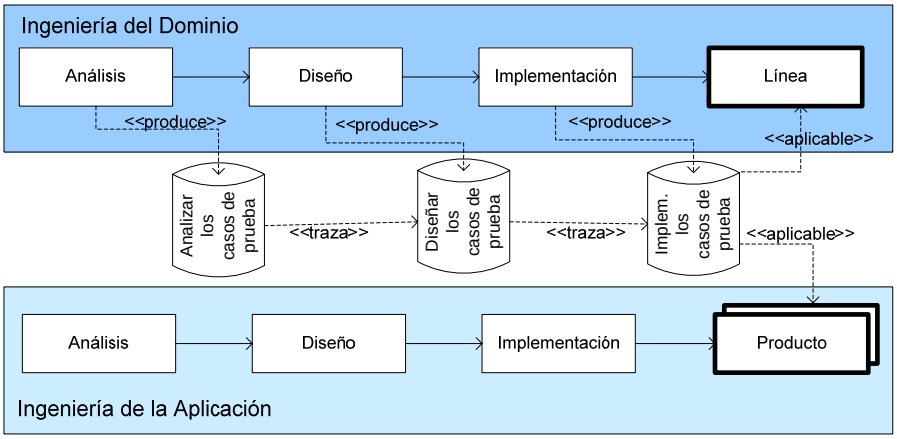

Figura 1. Proceso de desarrollo para líneas de producto software

La Figura 1 ilustra la forma en que se puede tomar ventaja de la trazabilidad para construir y reutilizar casos de prueba: suponiendo por simplicidad un modelo clásico de cascada, para cada etapa de la Ingeniería del Dominio es posible producir casos de prueba que pueden ser trazados de uno a otro nivel de abstracción (es decir, del Análisis hacia el Diseño), y además, para los nuevos artefactos producidos en cada etapa, se generan nuevos casos de prueba, que progresivamente, van enriqueciendo el conjunto de pruebas (es decir, casos de prueba producidos a partir de modelos de diseño completan los casos de prueba producidos desde el Análisis). Al final, todo este testware debe aplicarse a cada uno de los productos de la línea, y a la línea en sí.

Para hacer frente a un proceso bien gestionado, que permita la derivación de casos de prueba a partir de otros previamente definidos, y a partir de artefactos del análisis o diseño, es necesario el uso de herramientas y métodos estandarizados de la Ingeniería de Software.

Las pruebas dirigidas por modelos (Model-driven testing) requieren la derivación sistemática y en lo posible automatizada de las pruebas a partir de modelos [1]. En este artículo se presenta una estrategia para la generación de casos de prueba para LPS donde se define cómo gestionar la variabilidad en los artefactos de prueba y su trazabilidad al resto de los artefactos de desarrollo de la LPS. El trabajo aquí presentado es la continuación de investigaciones anteriores [12, 13]. En [13] se describe cómo se pueden generar modelos de prueba basados en el Perfil de pruebas de UML (UML-TP)[9] en forma automática a partir de las funcionalidad descritas como diagramas de secuencia para desarrollo de software convencional, utilizando como lenguaje de transformación entre modelos Query/View/Transformation (QVT)[10]. En [12] se presentan las extensiones al diagrama de secuencia y al UML-TP para gestionar la variabilidad en los modelos de prueba para LPS. En la propuesta presentada aquí se describe cómo las transformaciones entre modelos pueden extenderse a las pruebas en LPS, resolviendo la trazabilidad mediante el Modelo de Variabilidad Ortogonal (OVM) [14] para gestionar la variabilidad, mostrando un ejemplo de su uso en la LPS de Juegos de Mesa. El artículo se organiza de la siguiente manera: la sección 2 resume los trabajos relacionados, la sección 3 describe la Línea de Producto de Software de Juegos de Mesa donde se pone en práctica la propuesta, la sección 4 presenta la propuesta para generación de casos de prueba en LPS. Finalmente, en la sección 5 se presentan las conclusiones.

2. Trabajos relacionados

En general, las propuestas para la derivación de casos de prueba en LPS utilizan como base para el modelado UML ó artefactos de UML. Todas ellas contemplan la trazabilidad entre la Ingeniería del Dominio y del Producto en LPS. Sin embargo, se diferencian de la propuesta presentada aquí en que no toman en cuenta las capacidades de los marcos de modelado estándares específicamente diseñados para las pruebas tales como el Perfil de Pruebas de UML[9] ni utilizan lenguajes de transformación entre modelos como QVT. Una descripción completa de los trabajos existentes sobre pruebas en LPS puede encontrarse en [11], a continuación se resumen brevemente los trabajos que definen metodologías para la derivación de casos de prueba en LPS.

Nebut et al. [7] obtienen casos de prueba a partir de diagramas de secuencia de alto nivel, que luego se utilizan para generar automáticamente casos de prueba para cada producto. Bertolino et al. [2] proponen la metodología PLUTO (Product Line Use Case Test Optimization) que extiende la descripción textual de los casos de uso con un conjunto de etiquetas de variabilidad que son usadas para luego derivar los casos de prueba de la LPS. Kang et al. [4] extiende la notación del diagrama de secuencia para representar los escenarios de los casos de uso con variabilidad. Reuys et al. [15] presentan ScenTED (Scenario-based Test case Derivation), donde el modelo de pruebas (representado con diagramas de actividad) se construye a partir de las funcionalidades y usan diagramas de secuencia para representar el escenario de prueba. Olimpiew el al. [8] propone el método PLUS (Product Line UML-based Software engineering) de tres fases: creación del diagrama de actividad a partir de los casos de uso, creación de tablas de decisión a partir de los diagramas de actividad y creación de plantillas de prueba a partir de las tablas de decisión.

Dado que nuestro trabajo utiliza el Perfil de Pruebas de UML 2.0 (UML Testing profile, UML-TP) [9], a continuación se resume brevemente. El UML-TP extiende UML 2.0 con conceptos específicos para las pruebas, agrupándolos en: arquitectura de pruebas, datos de pruebas, comportamiento de pruebas y tiempos de prueba[9]. La arquitectura de las pruebas contiene la definición de todos los conceptos necesarios para realizar las pruebas. En ellas se definen el contexto de las pruebas y el resto de los elementos necesarios para definir las pruebas. El comportamiento de las pruebas especifica las acciones y evaluaciones necesarias para la prueba. El caso de prueba es el concepto principal en el modelo de prueba, y su comportamiento puede ser descrito usando el concepto Behavior de UML 2.0, diagramas de secuencia, máquinas de estado o diagramas de actividad. En el Perfil, un caso de prueba (test case) es una operación de un contexto de prueba que especifica cómo un conjunto de componentes cooperan con el sistema bajo prueba (system under test, SUT) para alcanzar el objetivo de prueba [1].

3 Ejemplo: Línea de Producto de Juegos de Mesa

Esta sección describe un resumen del caso de estudio, se trata de un sistema distribuido cliente-servidor donde jugar a uno o más juegos de una familia de juegos de mesa. Este tipo

de juegos comparten un amplio conjunto de características, tales como la existencia de un tablero, son jugados por uno o más jugadores, utilizan dados, existe la posibilidad de robar piezas, pueden incluir preguntas al jugador o políticas relacionadas con el paso del turno al siguiente jugador, etc. Aunque la mayoría de software desarrollado con LPS corresponde a aplicaciones empotradas [5, 6], este ejemplo es adecuado para ejemplificar cómo este paradigma relativamente nuevo puede ser aplicado también para el desarrollo de sistemas de software puro. La LPS permite cuatro tipos de juegos de mesa: Ajedrez, Damas, Parchís y Trivial. Dado que no es posible mostrar la LPS completa, en la Figura 2 se muestran algunos de los puntos de variación y sus variantes siguiendo la notación OVM [14]. En la notación gráfica de OVM, los puntos de variación están representados por triángulos y sus variantes con un rectángulo. Las líneas punteadas representan las variantes opcionales (es decir, pueden ser omitidas en algunos productos), mientras que las líneas continuas representan variantes obligatorias (que están presentes en todos los productos). Las asociaciones entre las variantes pueden ser: requires_V_V y excludes_V_V, dependiendo de si una variante requiere o excluye a otra. Del mismo modo, las asociaciones entre una variación y un punto de variación puede ser: requires_V_VP y excludes_V_VP, donde se indica si una variación requiere o excluye un punto de variación. La LPS de juegos de mesa tiene cuatro puntos de variación (ver Figura 2):

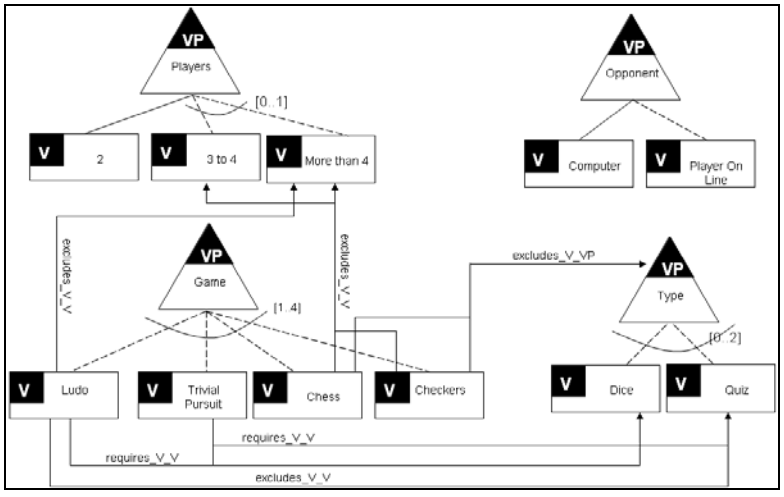

Figura 2. Puntos de variación y sus variantes para la LPS de Juegos de Mesa

- Juego (Game): Este punto de variación tiene cuatro variantes: Parchís (Ludo), Damas (Checkers), Ajedrez (Chess) y Trivial.

- Oponente (Opponent): El jugador puede jugar contra el ordenador (computer), o contra otro jugador que se encuentre en línea (Player On Line).
- Jugadores (Players): El número mínimo de jugadores para todos los juegos es 2, pero, algunos juegos se puede jugar de 3 a 4 o más de 4. El Ajedrez excluye estas dos opciones, mientras que el Ludo excluye la opción de más de 4 jugadores.
- Tipo (Type): Algunos juegos utilizan dados (Dice) o realizan preguntas al jugador (Quiz). El Trivial utiliza ambas opciones y el Ajedrez y las Damas excluyen este punto de variación. El Ludo requiere los dados pero excluye las preguntas.

4. Generación de casos de prueba para Líneas de Producto Software

La generación automática de los casos de prueba para Líneas de Producto de Software requiere definir los mecanismos para gestionar la variabilidad. Los artefactos de diseño de la LPS contienen la variabilidad que permite luego derivar cada producto. La trazabilidad de dicha variabilidad a los modelos de prueba, ayuda al reuso de las pruebas y a la mantenibilidad de la LPS. Es necesario definir cómo se trazará la variabilidad desde los artefactos de diseño del dominio de la LPS a los artefactos de prueba. En nuestra propuesta, para la generación automática de casos de prueba en LPS se utilizan modelos que definen las funcionalidades de la LPS como entrada y se generan automáticamente los modelos de prueba que contienen los casos de prueba para esas funcionalidades. Las transformaciones entre modelos se realizan utilizando el lenguaje QVT, la Figura 3 muestra los modelos fuente y destinos en la transformación. En la Ingeniería del Dominio de la LPS, el diagrama de secuencia UML con variabilidad y el modelo de variabilidad OVM se transforman mediante QVT en los siguientes modelos: Modelo de Variabilidad OVM enriquecido con las relaciones a los elementos de prueba, Diagrama de secuencia describiendo el caso de prueba y Arquitectura de Pruebas.

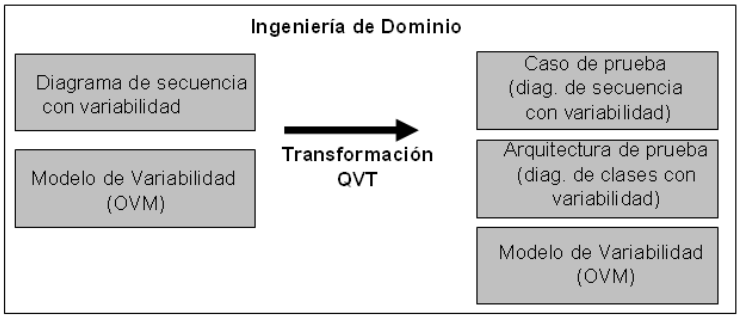

Figura 3. Generación de las pruebas usando QVT

Se utilizan los siguientes modelos para gestionar la variabilidad a nivel de la ingeniería de Dominio en LPS:

- Diagrama de secuencia con variabilidad: Este modelo describe un escenario de un caso de uso representado como un diagrama de secuencia UML. La variabilidad se representa utilizando el estereotipo *Variation Point* en el *CombinedFragment* del diagrama de secuencia UML. El diagrama de secuencia con variabilidad se utiliza tanto para describir la funcionalidad a probar, como para describir el comportamiento del caso de prueba para esa funcionalidad.
- Modelo Ortogonal de Variabilidad: El Modelo de Variabilidad Ortogonal (OVM) [14] contiene los datos sobre los puntos de variación (VP) y sus variantes. OVM permite representar las dependencias entre puntos de variación y elementos variables, así como las asociaciones entre los VP y las variantes con los artefactos de desarrollo. Dado que nuestra propuesta se encuentra enmarcada dentro de UML, fue necesario definir el Modelo Ortogonal de Variabilidad como Perfil de UML.
- Arquitectura de pruebas: Es un diagrama de clases donde se describen los elementos del UML-TP que se requieren para generar el caso de prueba.

La Figura 4 muestra dos modelos de la LPS de Juegos de Mesa, el modelo de arriba es un Modelo de Variabilidad conforme con el Perfil de OVM que representa los puntos de variación Juego (Game) y Tipo (Type), los cuales están estereotipados como VariationPoint. Las variantes están estereotipadas como Variant. Las asociaciones entre los puntos de variación y sus variantes están estereotipadas como opcional (*optional*) y las restricciones entre los elementos también se muestran como asociaciones estereotipadas como *requires* o *excludes*. La ortogonalidad del modelo puede verse en las asociaciones

entre el modelo de variabilidad (parte de arriba) y el diagrama de secuencia (parte de abajo), dicha asociación está estereotipada como realized by.

El modelo de variabilidad representado como Perfil de UML contiene la misma información que la notación gráfica de OVM de la Figura 2, la diferencia es que el Perfil UML para OVM nos permite asociar los puntos de variación y las variantes a cualquier artefacto de UML. En el caso del ejemplo vemos que las variantes Ludo, Trivial y Dice están relacionadas con el *CombinedFragment* del diagrama de secuencia. Es posible porque en el Perfil UML para OVM se asocian con el concepto *Element* de *UML MetaClass*.

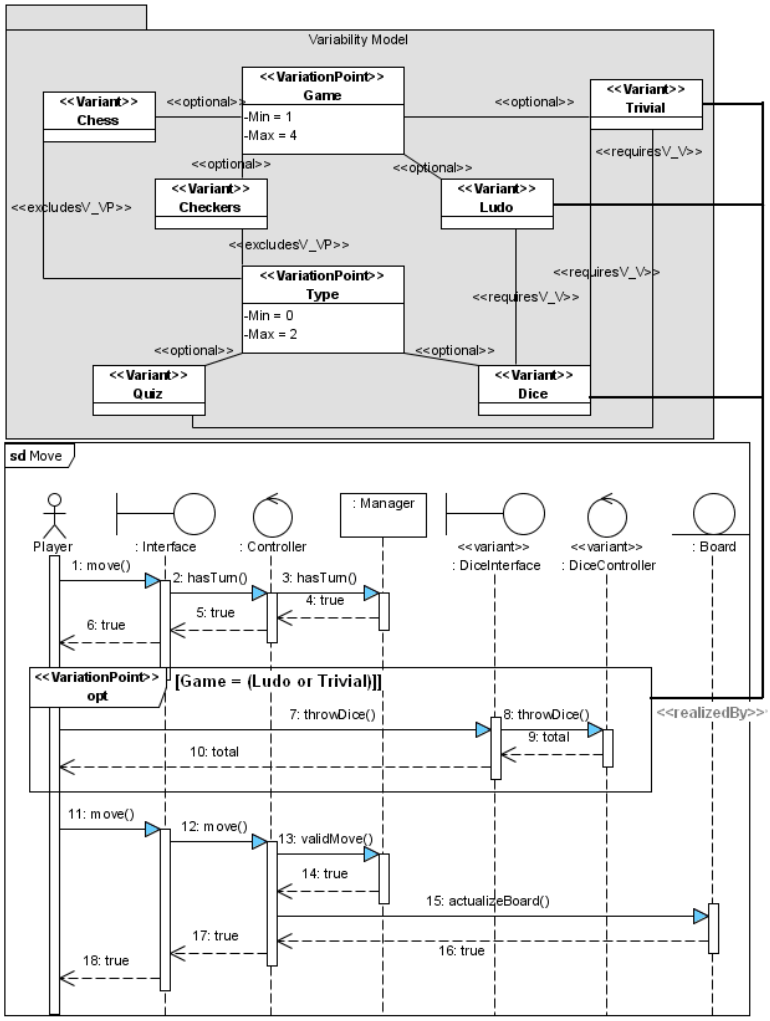

Figura 4. Diagrama de secuencia de Mover Pieza y su relación con el Modelo OVM

El segundo modelo representado en la Figura 4 es un diagrama de secuencia, que utiliza una extensión de las *Interaction* de UML para LPS, dicha extensión se basa en agregar el estereotipo *Variation Point* al *CombinedFragment* en los diagramas de secuencia. En el

ejemplo de la Figura 4, el diagrama de secuencia representa la funcionalidad de Mover en un juego de mesa. Para esto, el jugador expresa su intención de mover la pieza y el sistema comprueba que tenga el turno. El *CombinedFragment* estereotipado como *Variation Point* es quien comprueba si el juego es Ludo o Trivial, en dicho caso el jugador debe tirar los dados antes de mover. Para los demás juegos (Ajedrez y Damas), esta funcionalidad se ignora y no forma parte del producto. Luego, el jugador mueve la pieza y se actualiza el tablero. Una descripción completa de la extensión definida para manejar la variabilidad en los diagramas de secuencia puede consultarse en [12].

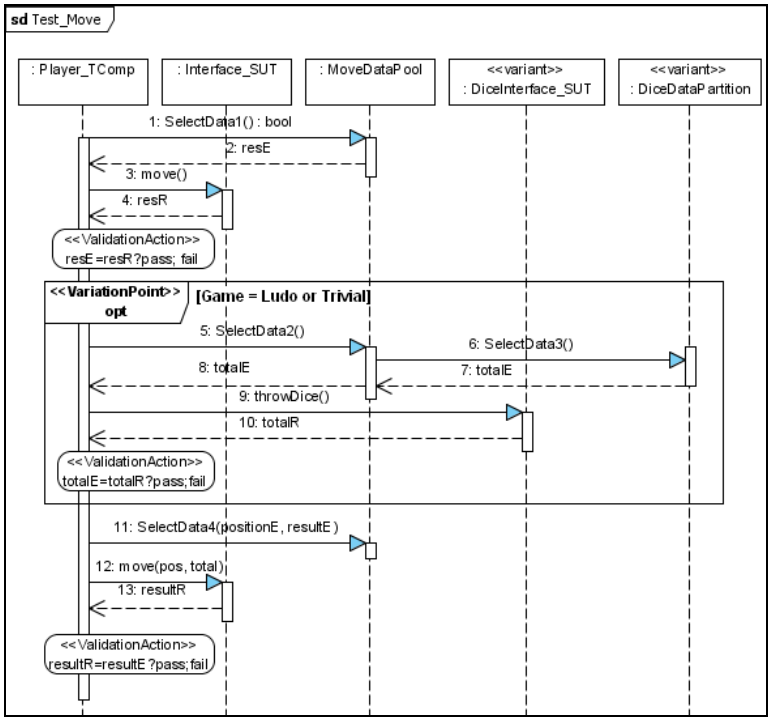

Figura 5.Caso de prueba para la funcionalidad "Mover pieza"

A partir de los modelos fuente de la Figura 4, se debe generar automáticamente mediante QVT el caso de prueba y la arquitectura de prueba. Además el modelo OVM debe ser actualizado con la trazabilidad de la variabilidad a los artefactos de prueba. El comportamiento del caso de prueba conforme con el Perfil de Pruebas UML se muestra en la Figura 5 y cumple los siguientes pasos:

- Obtener los datos de prueba: El *testComponent Player_TComp* invoca el *DataSelector* en el *dataPool* que retorna los datos necesarios para probar la función.
- Ejecutar el caso de prueba en el SUT: el *testComponent Player_TComp* simula al actor y éste, a su vez, invoca la función a probar en el *Interface_SUT.*

- Obtener el resultado del caso de prueba: El *testComponent* es el responsable de comprobar si el resultado retornado por el SUT es correcto, para esto utiliza las acciones de validación (*ValidationActions*) que son las encargadas de informar al árbitro (*arbiter*) el resultado de la prueba.

Dado que el diagrama de secuencia que vamos a probar tiene un *CombinedFragment* etiquetado *Variation Point* (ver Figura 4), puede verse en la Figura 5 que la prueba de esa porción de la funcionalidad también se realiza dentro de un *CombinedFragment* etiquetado *Variation Point.* Para cada *interaction operand* dentro del *CombinedFragment*, se realizan los mismos pasos descritos arriba: se obtienen los datos, se ejecuta la funcionalidad en el SUT y se obtiene el veredicto de la prueba. En el ejemplo se tiene un solo interaction operand con la guarda [Game=Ludo or Trivial], los datos de prueba son obtenidos invocando a *SelectData3()* que retorna el valor *totalE* que corresponde al resultado esperado para el caso de prueba. Luego, la operación *throwDice()* es invocada en el SUT llamado *DiceInterface_SUT*. El SUT retorna el resultado real *totalR*. Por ultimo, un *validationAction* compara el resultado esperado (*totalE*) con el resultado real (*totalR*) para obtener el veredicto. Han sido desarrolladas las transformaciones QVT que permiten generar los artefactos de prueba con variabilidad para LPS.

3. Conclusiones

Se ha presentado una propuesta para la generación automática de casos de prueba para LPS. La propuesta gestiona la variabilidad en los modelos de prueba y mantiene la trazabilidad con los modelos de diseño de la LPS a nivel de Ingeniería de Dominio. Como trabajo futuro se encuentra la derivación de los modelos de prueba a nivel de la Ingeniería de la Aplicación, lo que implica resolver la variabilidad para cada producto.

Agradecimientos

Este trabajo ha sido parcialmente soportado por los proyectos de la Junta de Comunidades de Castilla-La Mancha: PRALIN (PAC08-0121-1374) y MECCA (PII2I09-0075-8394). Se ha contado además con el apoyo de la Beca de Investigación. Orden de 13-11-2008 de la Consejería de Educación y Ciencia. Junta de Comunidades de Castilla-La Mancha (JCCM).

Referencias

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

[2] Bertolino, A., S. Gnesi, y A. di Pisa, PLUTO: A Test Methodology for Product Families. Software Product-family Engineering: 5th International Workshop, 2003.

[3] Clements, P.y L. Northrop, Salion, Inc.: A Software Product Line Case Study. 2002, DTIC Research Report ADA412311.

[4] Kang, S., et al., Towards a Formal Framework for Product Line Test Development. 7th IEEE International Conference on Computer and Information Technology, 2007.

[5] Kim, K., et al. ASADAL: a tool system for co-development of software and test environment based on product line engineering, 28th Inter. Conf. on Soft. Engin. 2006.

[6] Kishi, T.y N. Noda, Formal verification and software product lines. Communications of the ACM, 2006. 49(12): p. 73-77.

[7] Nebut, C., et al., Automated requirements-based generation of test cases for product families. 18th IEEE International Conference on Automated Software Engineering, 2003.

[8] Olimpiew, E.y H. Gomaa, Customizable Requirements-based Test Models for Software Product Lines. International Workshop on Software Product Line Testing, 2006.

[9] OMG, UML 2.0 Testing Profile Specification. 2004, Object Management Group.

[10] OMG, Meta Object Facility 2.0 Query/View/Transformation Specification, v1.0. 2007.

[11] Pérez Lamancha, B., M. Polo Usaola, y M. Piattini. Software Product Line Testing, A systematic review, 4th International Conference on Software and Data Technologies, 2009.

[12] Pérez Lamancha, B., M. Polo Usaola, y M. Piattini. Towards an Automated Testing Framework to Manage Variability Using the UML Testing Profile, Workshop on Automation of Software Test. IEEE Catalog CFP0915D, ISBN978-1-4244-3711-5. Vancouver, Canadá, 2009.

[13] Pérez Lamancha, B., et al., Propuesta para pruebas dirigidas por modelos usando el perfil de pruebas de UML 2.0. REICIS, 2008. 4(4): p. 28-41. ISSN: 1885-4486.

[14] Pohl, K., G. Böckle, y F. Van Der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques. 2005: Springer.

[15] Reuys, A., et al., Model-based System Testing of Software Product Families. Advanced Information Systems Engineering, CAiSE, 2005.

Análisis de la calidad y productividad en el desarrollo de un proyecto software en una microempresa con TSPi

Edgar Caballero
Universidad Politécnica de Madrid, Facultad de Informática, Dpto. Lenguajes y sistemas informáticos e ingeniería del software
ecaballero@bolesfactory.com
Jose Antonio Calvo-Manzano, Gonzalo Cuevas, Tomás San Feliu
Universidad Politécnica de Madrid, Facultad de Informática, Dpto. Lenguajes y sistemas informáticos e ingeniería del software
{jacalvo, gcuevas, tsanfe}@fi.upm.es

Resumen

Este artículo presenta el análisis de los resultados obtenidos al aplicar TSPi en el desarrollo de un proyecto software en una microempresa desde el punto de vista de la calidad y la productividad. La organización en estudio necesitaba mejorar la calidad de sus procesos pero no contaba con los recursos económicos que requieren modelos como CMMI-DEV. Por esta razón, se decidió utilizar un proceso adaptado a la organización basado en TSPi, observándose una reducción en la desviación de las estimaciones, un incremento en la productividad, y una mejora en la calidad.

Palabras clave: TSPi, Microempresas, Calidad de software, Mejora de procesos

Analysis of quality and productivity with TSI in a software development Project in a microcompany

Abstract

This article shows the benefits of developing a software project using TSPi in a "Very Small Enterprise" based in quality and productivity measures. An adapted process from the current process based on the TSPi was defined and the team was trained in it. The workaround began by gathering historical data from previous projects in order to get a measurement repository, and then the project metrics were collected. Finally, the process, product and quality improvements were verified.

Key words: TSPi, Very Small Enterprise, Software Quality, Process improvement.

Caballero, E., Calvo-Manzano, J.A., Cuevas, G. y San Feliu, T." Análisis de la calidad y productividad en el desarrollo de un proyecto software en una microempresa con TSPi", REICIS, vol. 5, no.2, 2009, pp.28-37. Recibido: 22-6-2009; revisado: 6-7-2009; aceptado: 31-7-2009

1. Introducción

El principal problema de una microempresa de desarrollo de software es la falta de recursos para poder invertir en la definición y mejora de sus procesos, descuidando aspectos claves

como son la gestión y la calidad [1]. Pero esto no es reciente y se presenta en todo tipo de organizaciones observándose proyectos desbordados en coste y tiempo, con baja calidad y un alto índice de cancelaciones [2].

Han aparecido modelos de procesos como el CMMI-DEV que han tenido bastante éxito, pero su implementación operativa representa una fuerte inversión económica y es de gran complejidad para las organizaciones pequeñas [1].

Actualmente existe una gran preocupación por esta situación y se están realizando investigaciones sobre la mejora de procesos en entornos de microempresa [3] para poder dotar a este sector de habilidades y capacidades cruciales en un mercado globalizado [4].

Una alternativa para poder alcanzar niveles de madurez CMMI en microempresas es mostrada en [5] y [6] a través del uso de TSP (Team Software Process). Su efectividad es tal que puede mejorar la calidad de proyectos en organizaciones con CMMI Nivel 5 [7].

TSP es una definición de proceso operativa que provee un equilibrio entre proceso, producto y equipo de trabajo, basado en una amplia experiencia en planificación y gestión de proyectos software [8].

En el presente artículo se muestra, a través de un caso de estudio, la experiencia de haber empleado un proceso adaptado a la organización basado en la estrategia introductoria del Team Software Process (TSPi).

El objetivo del artículo consiste en analizar este proceso adaptado considerando la calidad y la productividad del mismo en una microempresa, a través de la verificación de los siguientes objetivos:

Número	Objetivo
Objetivo 1	Reducir la desviación de las estimaciones
Objetivo 2	Verificar la mejora de la productividad
Objetivo 3	Verificar la mejora de la calidad y del proceso

Tabla 1. Objetivos del Proyecto

El proyecto bajo estudio tenía un calendario y presupuesto restringidos, y se temía no poder cumplir ni con los plazos de entrega ni con los costes establecidos aplicando el proceso actual de la organización. Por otro lado, se trataba de un cliente importante al cual se debía atender su solicitud. En estas circunstancias, y consientes del riesgo que implicaba

utilizar un nuevo proceso, se decidió adaptar el proceso actual aplicando las mejores prácticas adquiridas y las propuestas por TSPi para tratar de garantizar los tiempos y la calidad del producto, asumiendo el riesgo que esto implicaba.

El esquema representado en la Figura 1 resume el trabajo realizado en el artículo, el cual consistió en definir un nuevo proceso adaptado en base al propuesto por TSPi y al que actualmente utilizaba la organización. Para poder realizar la verificación de los objetivos del proyecto se procedió a analizar y comparar los datos recolectados de proyectos anteriores y las medidas obtenidas durante el desarrollo del proyecto piloto.

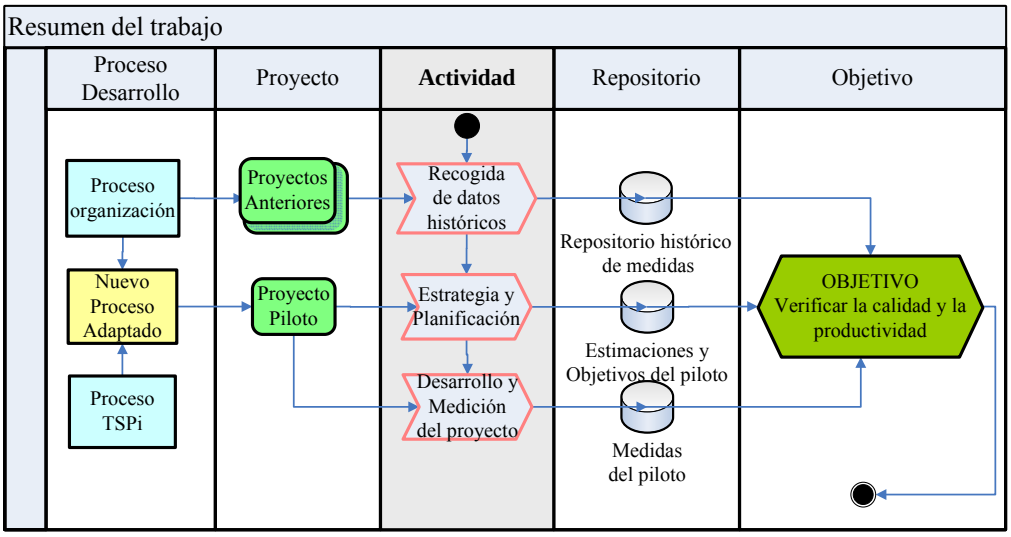

Figura 1. Resumen del trabajo realizado.

2. Escenario: La organización y el proyecto

BolesFactory es una microempresa española de desarrollo de software que dispone de un centro off-shore en Latinoamérica, la cual debido al incremento de sus proyectos se ha situado en un nuevo escenario donde existen diversos desarrollos ejecutándose a la vez, y con un mayor número de implicados.

En una evaluación interna, la directiva detectó que los proyectos estaban comenzando a retrasarse y a superar los costes establecidos, teniendo que emplear un esfuerzo adicional para cumplir los objetivos.

La conclusión de esta evaluación, determinó que los procesos empleados no eran efectivos en el nuevo escenario. La organización se interesó en incorporar algún modelo de

procesos como CMMI-DEV pero no contaba con los recursos ni con los conocimientos necesarios. Finalmente, los resultados de la evaluación impulsaron la idea de adaptar el TSPi al proceso de la empresa y aplicarlo al proyecto de estudio.

Antes de comenzar el proyecto, se realizó una formación en el proceso adaptado, y se procedió a recolectar datos históricos de proyectos anteriores para facilitar la estimación y tener un referente comparativo al determinar la efectividad del proceso.

3. Recolección de datos históricos

Los datos de la organización sobre proyectos anteriores no eran suficientes para nuestro estudio. Solamente existía información del calendario y coste, pero además se necesitaban datos sobre defectos o esfuerzo por fases para verificar todos los objetivos.

Ante esta circunstancia, los defectos y el esfuerzo fueron obtenidos a través de un recuento aproximado mediante la revisión de actas de reunión, informes, registros de incidencias y memorias técnicas de los proyectos históricos más significativos.

Para facilitar el recuento, los proyectos se dividieron en 3 fases (ver tabla 2):

Fase	Alcance
Desarrollo	Desde el lanzamiento del proyecto hasta las pruebas unitarias
Pruebas	Abarca el periodo de pruebas de integración y del sistema
Entrega	Desde la entrega del producto al cliente hasta los primeros 3 meses de uso

Tabla 2. Fases del proyecto

Los proyectos históricos seleccionados fueron: HIS-1 (104 KLOC), HIS-2 (33 KLOC), HIS-3 (23 KLOC), HIS-4 (11 KLOC) e HIS-5 (7 KLOC). Estos proyectos son los más representativos para la empresa porque cada uno de ellos caracteriza los diferentes tipos de aplicaciones desarrolladas por la organización.

4. Proceso utilizado

El proceso utilizado en el proyecto es un proceso adaptado que combina los principios básicos del TSPi con el proceso interno de la organización. Una vez se definió el nuevo proceso, se impartió una formación y se inició el proyecto con la reunión de lanzamiento.

La principal diferencia en relación al proceso anterior es la importancia que se da a la calidad desde el principio del proyecto. TSPi es un proceso centrado en la reducción temprana de defectos, e incorpora en el plan de calidad aspectos de rendimiento de procesos y fases, revisiones formales, inspecciones y recuento de defectos.

Para la gestión del proyecto, se incorporaron las reuniones semanales y la monitorización del plan del proyecto en base al método del valor ganado. En la planificación de las tareas se establecieron planes semanales, lo que ayudó con respecto a la visibilidad del avance en el proyecto y su respectiva monitorización (ver figura 2).

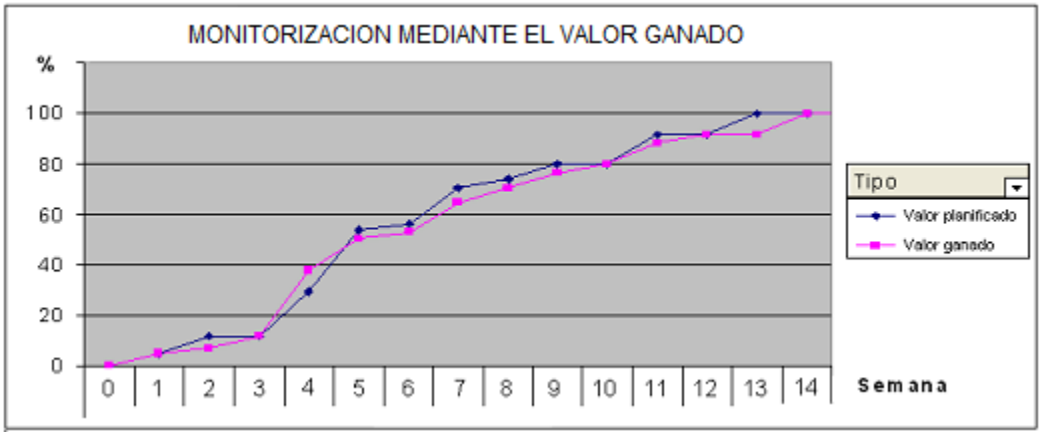

Figura 2. Monitorización del proyecto mediante el valor ganado.

Las reuniones semanales tenían por objetivo evaluar el calendario, los objetivos, los posibles riesgos, y la gestión de requerimientos.

En la Tabla 3 se resumen los principios básicos de TSPi que fueron aplicados.

Nuevo proceso	Anterior proceso
Proceso y roles bien delimitados	Proceso ambigüo con fases mal delimitadas.
Filosofía de equipo: colaboración y compromiso	Centrado tareas asignadas por el jefe proyecto
Calidad basada en la reducción temprana de defectos	Ante calendarios ajustados la calidad era relegada
Introducción de inspecciones formales en el proceso	Solo revisiones personales sin control de calidad
Delimitación realista y detallada del alcance	Se aceptan costes y tiempos sin definir el alcance
Monitorización del proyecto en base al valor ganado	No existía ningún mecanismo de monitorización
Reuniones semanales de seguimiento	No existían reuniones formales frecuentes

Tabla 3. Principios TSPi aplicados al nuevo proceso

5. Efectividad del nuevo proceso

Para poder verificar la mejora del nuevo proceso, se analizaron las estimaciones, la productividad y la calidad.

5.1. Objetivo 1: Reducir la desviación de las estimaciones

Como se observa en la Tabla 4, la desviación en las estimaciones se ha reducido. El porcentaje de reducción alcanzada se ha calculado a través de la siguiente fórmula:

% Reducción = 100 * (PRO – Media Histórica)/Media Histórica

Objetivo	Media histórica	PRO	% Reducción
% Desviación calendario	21,4%	7,7%	- 64,0%
% Desviación esfuerzo	55,9%	18,00%	- 67,8%
% Desviación tamaño	33,7%	22,6%	- 32,9%

Tabla 4. Fases del proyecto

La desviación obtenida en el proyecto piloto todavía es considerable, pero lo importante es la mejora alcanzada en el proceso (ver figura 3).

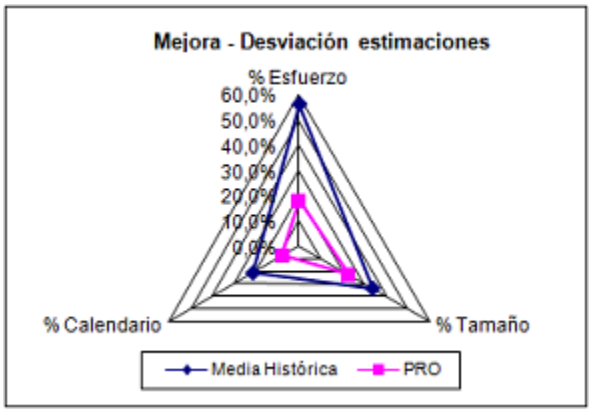

Figura 3. Desviación de las estimaciones en relación a la media histórica.

5.2. Objetivo 2: Verificar la mejora de la productividad

Para realizar esta verificación se ha analizado la densidad (ρ) de defectos a la entrega (Número de defectos eliminados en el momento de la entrega del producto por cada mil líneas de código), la productividad de las pruebas y la productividad del proceso de desarrollo.

Objetivo	Media histórica	PRO	Reducción

ρ Defectos a la entrega [def./KLOC]	2,0	1,2	- 40,0%
Prod. Pruebas [hra/KLOC]	33,4	13,2	- 60,5%
Prod. Proceso de desarrollo [hra/KLOC]	7,3	7,6	+ 4,1%

Tabla 5. Fases del proyecto

De los tres elementos mostrados en la Tabla 5, el que ha mejorado considerablemente es la productividad en las pruebas debido a la reducción de defectos antes de entrar en esta fase, gracias a actividades de revisión e inspección que sugiere TSPi en las fases anteriores.

5.3. Objetivo 3: Verificar la mejora de la calidad y del proceso

Para verificar la mejora de la calidad y determinar el grado de efectividad del proceso, se analizó la densidad de defectos en cada fase (Número de defectos eliminados al entrar a una fase por cada mil líneas de código). Una fase tiene calidad aceptable cuando la siguiente fase tiene una densidad de defectos menor o igual que la anterior [9].

Al ser un proceso centrado en la reducción temprana de defectos, TSPi propone llegar a la fase de pruebas con muy pocos defectos para que durante esta se logre identificar los restantes y se entregue al cliente un producto con la menor cantidad de defectos posible [9].

En la Figura 4 se analiza la densidad de defectos de cada fase y se observa que el nuevo proceso es más efectivo que el anterior porque la densidad de defectos se redujo conforme se avanzaba en el proceso.

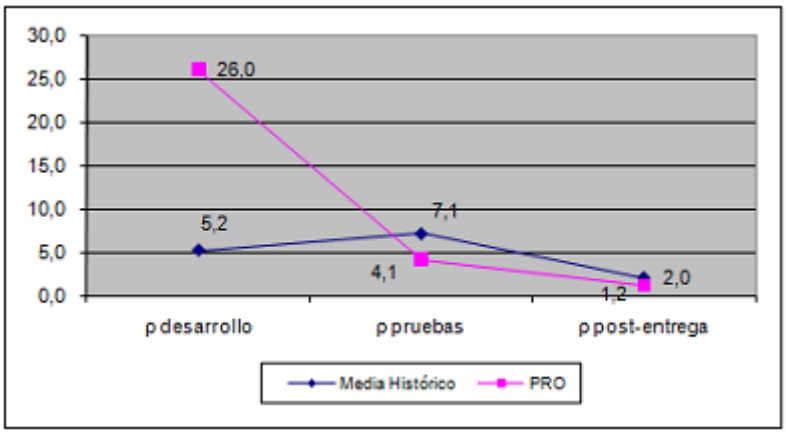

Figura 4. Mejora del proceso – Densidad de Defectos.

TSPi utiliza el "Rendimiento del proceso" (*Process Yield)* [9] para analizar la efectividad del proceso. La Figura 5 muestra el rendimiento del nuevo proceso centrado en la calidad, el cual alcanzó valores cercanos a los esperados por TSPi.

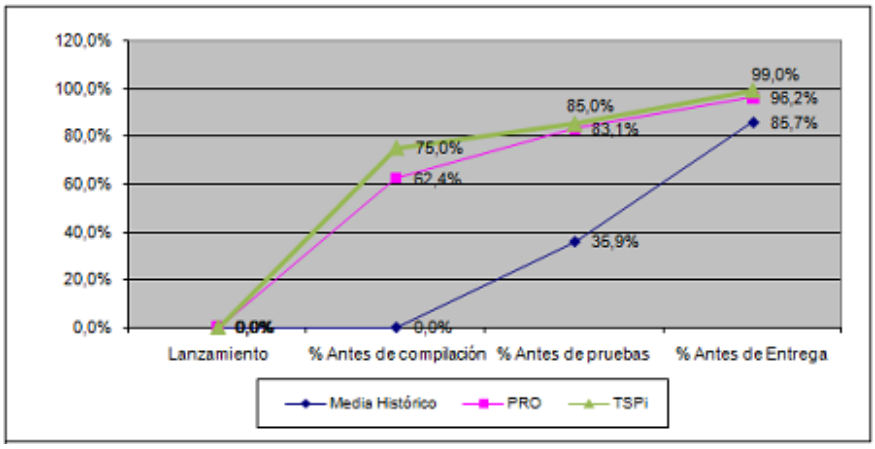

Figura 5. Mejora del proceso – Process Yield.

El coste de la calidad "COQ (*Cost of Quality*)" es otro aspecto importante a analizar. Tiene 3 componentes pero TSPi utiliza solamente dos: "*COQ = Appraisal + Failure*" [9]:

- "*Appraisal cost*": Coste de evaluar el producto y determinar su nivel de calidad. Incluye actividades de revisión e inspección.
- "*Failure cost*": Coste de diagnosticar un fallo y hacer las correcciones necesarias. Incluye actividades de compilación y pruebas.

La proporción del COQ no ha variado en el nuevo proceso (ver figura 6).

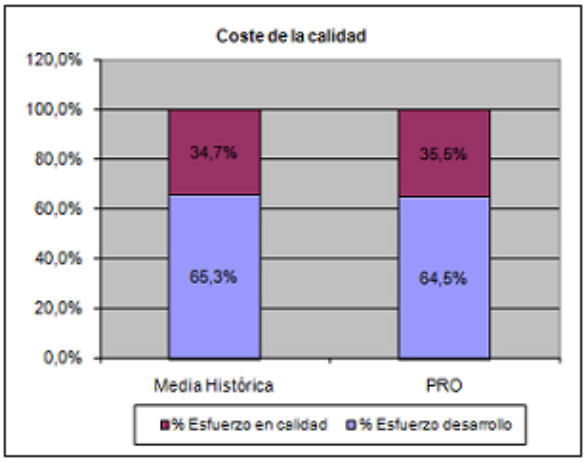

Figura 6. Mejora del proceso – Coste de la calidad.

A primera vista no se puede apreciar ninguna mejora, pero cuando se desglosa el coste de la calidad en sus dos componentes y se comparan entre sí, la mejora es evidente (ver figura 7).

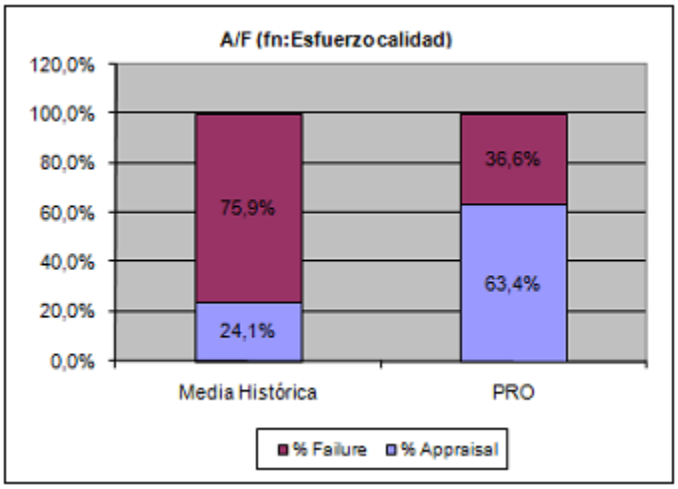

Figura 7. Mejora del proceso – Appraisal vs. Failure Cost.

6. Conclusiones

El uso de los principios de TSPi en el nuevo proceso ha permitido alcanzar los objetivos del proyecto en base a las siguientes consideraciones:

1. La integración del equipo, la delimitación del alcance funcional, el plan de proyecto detallado, la gestión de requerimientos, las reuniones semanales y la monitorización realizada con el método del valor ganado han incrementado la productividad y calidad del desarrollo.
2. Al aplicar la reducción temprana de defectos, la productividad en la fase de pruebas ha mejorado y el re-trabajo empleado ha disminuido.
3. Las actividades de revisión e inspección, y el plan de calidad, han permitido mejorar la calidad del producto. El equipo de desarrollo entendió la fase de pruebas como una actividad de evaluación de la calidad en lugar de una actividad de detección de defectos.

Sin necesidad de una inversión considerable, se ha demostrado que basarse en una definición de procesos operativa existente como es TSPi puede ser una solución de mejora intermedia para las microempresas.

Referencias

[1] International Process Research Consortium. IPSS White Paper, *Improving Process in Small Settings*, 2006. *http://www.sei.cmu.edu/iprc/ipss-white-paper-v1-1.pdf*

[2] Standish group. *2007 CHAOS Report*, 2007

[3] Garcia, S. Graettinger, C., Kost K., *Proceedings of the First International Research Workshop for Process Improvement in Small Settings*. SEI Special Report CMU/SEI-2006-SR-00, 2006.

[4] Glazer, H., *Time to Market vs. Process Discipline*. 2006
http://www.sei.cmu.edu/iprc/sepg2006/glazer.pdf

[5] Garcia, S., *Thoughts on Applying CMMI in Small Settings*. 2005 *http://www.sei.cmu.edu/cmmi/adoption/pdf/garcia-thoughts.pdf*

[6] Serrano, M., Montes, C., Cedillo, K., *An Experience on Implementing the CMMI in a Small Organization Using the TSP*, pp. 81-92, 2006.
http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06sr001.pdf

[7] Noopur, D., *The Team Software Process in Practice: A Summary of Recent Results*. SEI Technical Report CMU/SEI-2003-TR-014, 2003.

[8] Humphrey, W., *TSP: Coaching Development Teams*. Addison-Wesley, 2006.

[9] Humphrey, W., *Introduction to the Team Software Process*. Addison-Wesley

[10] Calvo-Manzano, J., Gonzalo, C., San Feliu, T., Caballero, E., *TSPi benefits in a software project under a Small Settings environment*. International Journal "Information Technologies and Knowledge" Vol.2 / 2008, pp. 245-250., 2008.

Asegurar que el software crítico se construye fiable y seguro

Patricia Rodríguez Dapena
SoftWcare SL
rodriguezdapena@softwcare.com

Resumen

El software falla y estos fallos pueden tener efectos catastróficos o afectar directamente a la fiabilidad y disponibilidad de los sistemas que utilizamos frecuentemente. Varios ejemplos ocurridos recientemente muestran que, incluso sistemas desarrollados siguiendo requisitos muy estrictos y fuertes medidas de control, fallan y causan daños irreparables (ej, pérdida de vidas humanas [9]). La seguridad 'safety' y la dependabilidad son características no-inherentes y son cada vez más importantes en más áreas y dominios. Hay que implementarlas en los sistemas y sub-sistemas explícitamente, como cualquier otro requisito. Existen diferentes técnicas y métodos que se utilizan para implementarlas, para los que no hay mucha experiencia en su utilización, ni aseguran el 100% de fiabilidad o de seguridad. El desarrollo de estos productos críticos resulta más caro, pero los riesgos de no desarrollarlos y mantenerlos adecuadamente son más caros aun: nos afectan a todos y no parece que haya nadie aun velando por nuestra seguridad y satisfacción. Este artículo presenta requisitos que exigen diferentes normas internacionales para implementar software crítico y analiza sus beneficios, inconvenientes y riesgos.

Palabras clave: Seguridad, dependabilidad, software crítico, SIL, niveles de criticidad, riesgo.

Assuring critical software is developed reliable and safe

Abstract

Software fails, and these failures can have catastrophic consequences o affect the reliability and availability of these critical systems we use everyday. Recent examples show that systems that have been developed following very strict requirements and strong control measures fail and may cause irreparable damage (e.g. loss of human lives [9]). Safety and dependability are non-inherent characteristics and they are very important in increasingly number of areas and daily live domains. They have to be consciously and explicitly implemented in systems and sub-systems, as any other requirement. In particular they are implemented using different techniques and methods, for which there is not much expertise and in practice use, and which do not ensure 100% reliability or security. The development of these critical software products is expensive, but the risks of not using the specific requirements and methods and techniques for their implementation are even more expensive: they affect all of us and it seems there is no one (yet) looking after us about our safety and satisfaction using them. This article presents safety and dependability requirements of different critical software-related standards and analyses some benefits, inconveniences and risks they have.

Key words: Safety, dependability, critical software, SIL, criticality levels, risk.

Rodríguez, P.,"Asegurar que el software se construye seguro y fiable", REICIS, vol. 5, no.2, 2009, pp.38-48. Recibido: 22-6-2009; revisado: 6-7-2009; aceptado: 31-7-2009

1. Introducción

El software en sí mismo, cuando falla no puede causar un accidente. Cuando se utiliza para controlar sistemas potencialmente peligrosos es cuando el software es crítico respecto a su seguridad (*safety*). El software en sistemas críticos se ha duplicado en pocos años en todas las facetas de nuestra vida cotidiana, además de ser cada vez más complejo, por lo que aumenta la posibilidad de que influya más directamente en la 'seguridad' y fiabilidad y disponibilidad de los sistemas que controla. En sistemas de consumo no considerados críticos respecto a su seguridad, la cantidad de software también se está duplicando casi cada año (ej, una televisión 'top-of-the-line' contiene software complejo cuando hace pocos años no tenía software). El usuario requiere cierto grado de disponibilidad y fiabilidad que no siempre se asegura.

Implementar, verificar y validar requisitos funcionales, operacionales, de interfaz es conocido y realizado en mayor o menor medida de una manera sistemática (el uso de la ISO 12207 [2] es cada vez más frecuente en los desarrollos software). Pero el concepto de implementar y verificar las características de seguridad *safety* y dependabilidad en un producto software aun no se realiza sistemáticamente en muchos campos donde sí se debería considerar crítico. Se requiere que los sistemas críticos con alto contenido de software funcionen correctamente en el entorno para el que han sido diseñados. El software que implementa funciones críticas del sistema, o que detecta peligros potenciales y/o que mitiga la consecuencia severa de los posibles accidentes, se exige que funcione correctamente (sea *dependable* o confiable) y que, además pueda reaccionar de manera segura y controlada incluso a cambios del entorno no esperadas. Además, el caso específico del software empotrado de control de sistemas críticos en tiempo real, del que no se tiene baja visibilidad de su funcionamiento en tiempo de ejecución, y para el que aun no hay tecnología para permitir su adecuada verificación, validación ni mantenimiento, son características mucho más complicadas de implementar.

Este artículo presenta brevemente diferentes requisitos de ciclo de vida del software crítico, diferentes mecanismos que se pueden/requieren utilizar y cuándo, lo imprescindibles que son, la poca experiencia existente en su aplicación, lo caro que puede resultar implementarlos, y los riesgos asumidos si se limita su uso a software crítico.

2. Software crítico y fallos de software

2.1 Seguridad y dependabilidad

Las características de dependabilidad y la seguridad se relacionan, aunque no son lo mismo. La seguridad *safety* se define como "esperanza de que un sistema en condiciones definidas no llega aun estado en el cual se ponga en peligro ni vidas humanas, ni la salud, ni las propiedades o el entorno" [8]. También se define como "estar libre de riesgos inaceptables" [3]. La seguridad (*safety*) se relaciona con el efecto final de cualquier fallo o evento. Es un aspecto a ser definido y controlado a nivel sistema. Los subsistemas y en particular el software que lo pueda controlar serán críticos según influyan en los aspectos de seguridad (*safety*) a nivel sistema. La dependabilidad en cambio se refiere a los fallos en sí mismos. En general, se define en diferentes estándares para dominios diferentes ([6], etc.) de diferente forma, pero se contabiliza la frecuencia y cantidad de fallos en sí mismos, sin evaluar sus consecuencias peligrosas. El término colectivo usado para describir cómo es la disponibilidad y los factores que la influencian: fiabilidad, mantenibilidad y soporte a la mantenibilidad [7]. Es a nivel sistema, subsistema o elemento. Por tanto, decir *software crítico* se puede referir a software cuyos aspectos de dependabilidad son importantes, o bien que tienen directa influencia en un sistema crítico respecto a su seguridad 'safety'.

De todos modos, es difícil imaginarse el desarrollar software crítico que sea seguro ('safe') pero no fiable. Si fallara mucho, pasaría a estado 'safe mode', sin hacer nada, y no sería un producto muy útil. Se necesita tener productos de software críticos seguros y fiables. Por tanto, aumentar la dependabilidad o la seguridad del software es una cuestión de enfoque en el control/eliminación o tolerancia de fallos del software y qué partes del software son las afectadas. El término criticidad se utilizará en este artículo como sinónimo de la seguridad y/o de la dependabilidad.

2.2 Software crítico

Un sistema puede realizar funciones que son críticas y entonces tiene que protegerse de eventos y/o fallos críticos para no causar daños importantes y estar disponible cuando se necesita. Para ello se diseñan dispositivos y/o mecanismos que mitiguen o controlen estos peligros y/o fallos. Estos eventos y/o fallos se representan en la Figura 1 como "Comportamiento inesperado" del sistema y los mecanismos para su control y mitigación corresponde a las zonas marcadas con un círculo como 'requisitos de seguridad y dependabilidad del sistema'. Tanto estos mecanismos de mitigación como otras funcionalidades críticas del sistema pueden ser implementados por software. Este software crítico debe funcionar correctamente, es decir, ni debe fallar en implementar las funciones críticas del sistema (debe controlar su propio comportamiento inesperado) ni debe ser la causa de tener que ponerse en funcionamiento los mecanismos de mitigación (esto corresponde a las zonas marcadas con un círculo como 'requisitos de seguridad y dependabilidad del software' en la [8]) y, además, no debe fallar en implementar mecanismos de mitigación de daños a nivel sistema que estén bajo su responsabilidad.

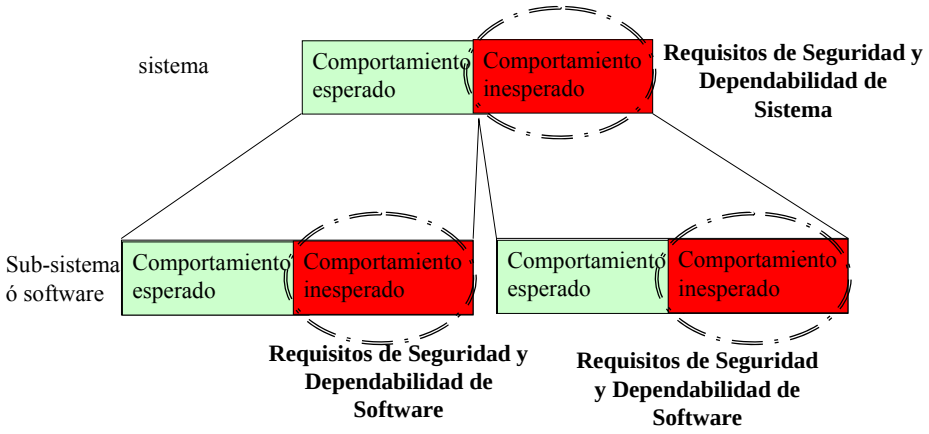

Figura 1. Requisitos y limitaciones de seguridad y dependabilidad del sistema y del software [1]

En general, el software crítico en un sistema crítico existe cuando:

- el software no falla de modo que cause o contribuya a un estado peligroso del sistema.
- el software no falla en la detección o la corrección estado peligroso del sistema.
- el software no falla en la mitigación o reducción del impacto del posible efecto si ocurre el accidente.

Los fallos de software son fallos sistemáticos puesto que se relacionan determinísticamente con una causa que sólo se puede eliminar modificando el diseño o código (o manuales de usuario/operaciones, etc.) [3]. El nivel de criticidad del software

depende únicamente del nivel de severidad del impacto que el fallo pueda ocasionar (el cálculo de las probabilidades de fallos de software es inmaduro y no se calcula en la práctica en la mayoría de los dominios para definir la criticidad).

La seguridad (*safety*) y dependabilidad de software están relacionados con los términos *failures*, *errors* y *faults*. Es necesario distinguir diferentes significados del término fallo de software, pues los mecanismos/medidas a implementarse para mitigar, prevenir estos 'fallos' de software se refieren a alguno de estos tres términos en particular:

1. *Failures* (fallo) se define como la terminación de habilidad de un elemento para realizar una función requerida [3]. Un software *failure* es la manifestación de un *error* [6].
2. *Error* es una discrepancia entre el valor o condición calculada, observada o medida y el valor o condición real, especificado o teóricamente correcto [3]. El error en el uso del software puede ser también la causa del *failure*.
3. *Fault* (fallo) es la causa de un *error* [4][6].

Implementar la dependabilidad y seguridad de software se centra en prevenir/eliminar/tolerar o predecir los fallos ('failure', 'error'y/o 'fault') de software.

3. Requisitos y procesos de ciclo de vida de software crítico

La seguridad y la dependabilidad del software, al ser características que hay que implementar, verificar, validar y evaluar explícitamente a través de la implementación de diferentes requisitos y mecanismos que se derivan del análisis de los requisitos y restricciones de seguridad y dependabilidad del sistema (ver Figura 1 anterior). Estos requisitos y mecanismos de deben planificar desde las primeras fases del ciclo de vida del software.

En cada una de las fases del desarrollo habrá que implementar, verificar y/o validar algún aspecto de estas características, como se muestra en la Figura 2. Existen cuatro grupos de métodos, para prevenir/eliminar/tolerar o predecir los fallos (*failure*, *error* y/o *fault*) de software.

Los tres primeros tipos de mecanismos son los más utilizados, mientras que el último aun es un grupo de métodos y técnicas inmaduras y aun en fase de investigación:

- Prevención de fallos - ¿cómo prevenir la aparición e introducción de fallos?

- Tolerancia a Fallos - ¿cómo proporcionar un servicio que cumpla con las especificaciones aún cuando se está produciendo un error?
- Eliminación de fallos - ¿cómo reducir la presencia de fallos, en relación con la cantidad y severidad de los fallos?
- Previsión de fallos - ¿cómo estimar la aparición de fallos?

Existen múltiples métodos y técnicas de cada tipo, y desgraciadamente no hay una lista estándar de cuáles utilizar en cada caso, ni qué combinación de técnicas es la mejor. Lo que sí es cierto es que la implementación de las características de seguridad y dependabilidad en el software puede resultar muy costosa dependiendo de los mecanismos, organización y requisitos que haya que implementar.

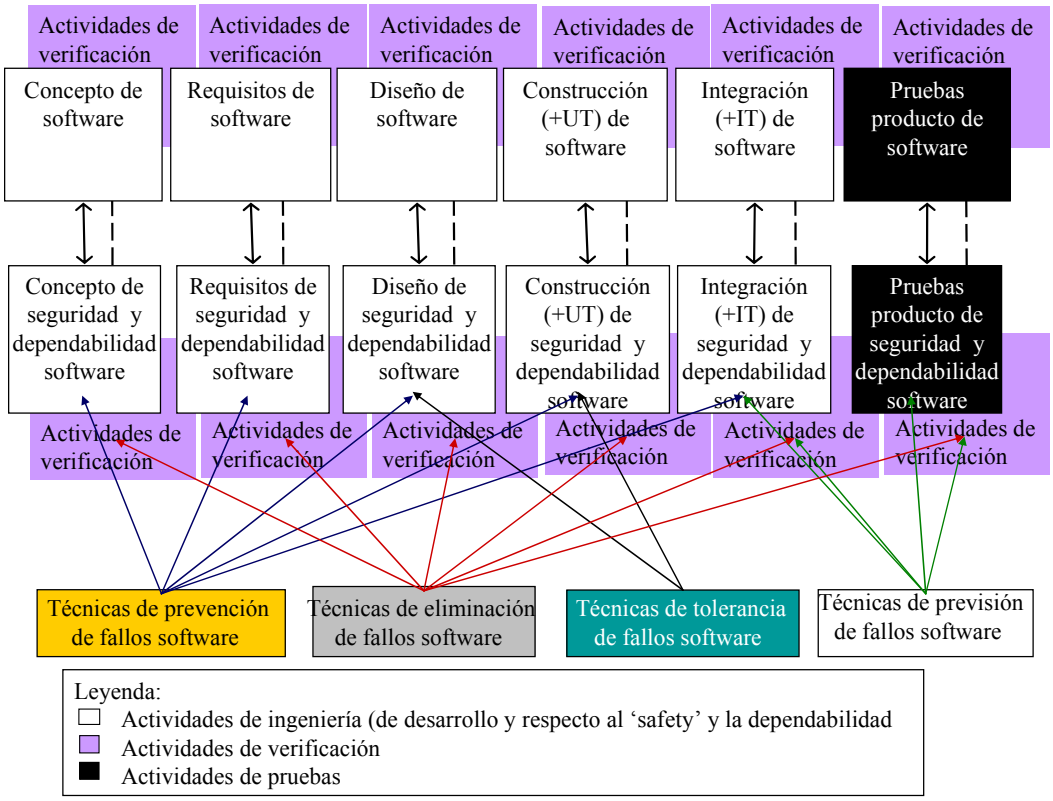

Figura 2. Mecanismos para la seguridad y dependabilidad del software 0

Los diferentes estándares internacionales existentes al respecto definen diferentes requisitos y limitaciones para la implementación de la seguridad o la dependabilidad del software y su aplicabilidad se requiere según lo que se denomina "nivel de criticidad" del software. El SIL o Nivel de Criticidad es un indicador del grado de confianza requerida para un producto o sistema, por lo que determinan qué medidas hay que poner para que, o

bien la función de mitigación sea fiable o que el fallo (*failure*) que pueda ocasionar un peligro no ocurra o se reduzca la severidad del efecto [8]. Los estándares donde se menciona la seguridad y/o la dependabilidad del software utilizan diferentes definiciones de niveles de criticidad e indicadores para definirlos. Además requieren:

- El uso de diferentes técnicas (prevención, tolerancia y/o análisis y eliminación) en las diferentes etapas del ciclo de vida del software para cada nivel de criticidad.
- Que las mismas actividades, técnicas y métodos se utilicen para el software configurable y sus datos, pues hay sistemas críticos que reutilizan software crítico genérico, pero configurable a través de sus datos, y éstos son críticos también.
- Diferentes niveles de independencia del personal para realizar algunas actividades, respecto del personal que lo desarrolla.
- Exigencias respecto a la confianza necesaria respecto a productos ya existentes que se quieren reutilizar como parte del software crítico.
- Exigencias respecto a las herramientas a ser utilizadas para el desarrollo, la verificación y la validación del software crítico, que pueden afectar a la calidad del producto final.

Estas diferencias se pueden ver al comparar estándares como el DO178B [4] (que contiene guías (utilizado en la práctica como requisitos) para la implementación y demostración de la seguridad-'safety' del software de control a bordo de los aviones, para las autoridades de certificación); o la norma NASA 8739.8 y su guía (NASA-GB-8719.8 [7]) (que contiene requisitos para el desarrollo de software crítico (seguridad) en sistemas espaciales para NASA); o la norma IEC 61508 [3] (que establece requisitos para todas las actividades relacionadas con el ciclo de vida de seguridad de los sistemas que incluyan componentes eléctricos y/o electrónicos y/o electrónicos programables (E/E/PES) que se utilizan para realizar las funciones de seguridad) y que es base para la futura ISO 26262 para dispositivos electrónicos en sector automoción; o la norma EN50128 [3], basada en la IEC 61508 0 antes mencionada, (especifica procedimientos y requisitos técnicos para el desarrollo de sistemas electrónicos programables para uso en aplicaciones de control y protección del ferrocarril); o la norma DEF-STAN-00-55 [5] (con requisitos de seguridad de software para sistemas de defensa del Reino Unido tiene requisitos tales como el uso de

lenguajes de descripción formal y verificaciones y pruebas formales para el software crítico).

Existen más ejemplos de estándares para software crítico donde se definen requisitos, métodos, técnicas, organizaciones, etc. para diferentes niveles de criticidad del software. La sección siguiente analiza los efectos de estos requisitos en las organizaciones y desarrollos software, además de los riesgos de no implementarlos.

4. Análisis de costes y riesgos asociados

Los diferentes requisitos para la implementación de la seguridad y dependabilidad del software en sistemas críticos requieren:

- Que los desarrolladores de software tengan conocimientos de los mecanismos específicos de seguridad y dependabilidad, tanto para la ingeniería (tolerancia a fallos y prevención de fallos - métodos más o menos formales para el desarrollo, uso de estándares limitados de programación, etc.), como para la verificación y análisis del software (eliminación de fallos). Es necesario además que sepan seleccionar cuáles son las técnicas a utilizarse y combinarse, valorando otros requisitos de rendimiento, tiempo real, ocupación de memoria, operacionales, etc. y, en la mayor parte de los casos, a ser propuestas y negociadas con el cliente. Por tanto, en las organizaciones:
    1. Posible formación necesaria
    2. Limitación del personal que podrá desarrollar estos productos
- Que las organizaciones planifiquen roles más o menos independientes para cada proyecto: a veces los que desarrollan deberán ser independientes de los que verifican, de los que validan y de los que evalúan la seguridad, por tanto:
    1. Más compleja organización aumentando la gestión de los proyectos
    2. Más compleja organización implica más formalidad en el proyecto, que puede conllevar a más duración y más esfuerzo
- Cambios en las herramientas que se utilicen para los desarrollos: no cualquier herramienta puede ser utilizada, por tanto:
    1. Compra y formación de nuevas herramientas de desarrollo, o
    2. Cualificación y pruebas de las herramientas existentes

- Que la reutilización de software ya existente se realice cumpliendo los mismos requisitos que el software crítico nuevo, requiere realizar actividades extra.

Por tanto, el desarrollo de software crítico es caro de implementar. Pero:

- ¿Qué riesgos se asumen y quién los asume si no se implementan los sistemas críticos con alto contenido en software con estos requisitos?
- ¿Cómo se asegura mayor fiabilidad y seguridad del software empotrado en los 100 ECUs de un automóvil cuando hoy día, en general, no se exige que demuestren el cumplimiento de este tipo de requisitos? (la norma ISO/IEC 26262 para seguridad funcional en vehículos de carretera aún no se ha publicado y todavía no es de obligado cumplimiento).
- ¿Cómo asegurar que los dispositivos médicos, cada vez más electrónicos (controlados por software), cumplan los más estrictos requisitos de dependabilidad y de seguridad?
- ¿Y el software del cajero automático? ¿y el software de control semafórico?...
- ¿Quién debe decidir qué requisitos son los que se implementarán y después se verificarán y validarán en estos sistemas críticos? ¿Quién será el responsable de su aseguramiento?

En muchos dominios, hoy día, son los fabricantes de estos sistemas los que deciden el riesgo a ser sumido por los usuarios. Estos riesgos, en muchas ocasiones, se miden por datos de mercado y ventas, más que por datos de riesgos de seguridad para los usuarios. Pero, por otro lado, ¿Se sabe el riesgo que asume al utilizar o depender de sistemas críticos controlados por software? Además, las reparaciones y mantenimiento de estos sistemas y de su software son cada vez más sofisticados. Hasta hace pocos años, los coches se reparaban en los talleres mecánicos y se conocían los mecanismos hidráulicos que tenían, o las lavadoras se arreglaban con 'chapucillas doméstica' que conseguían que siguiera funcionando. Ahora, con los subsistemas electrónicos, más sofisticados y complejos, ya no se puede corregir el software más que de los fabricantes directamente, que nos cambian las piezas completas (cajas 'negras') por nuevas que contienen nuevas versiones del software.

¿Quién asegura la seguridad y satisfacción de los usuarios de estos sistemas? ¿Es necesaria más regulación y legislación para exigir a los fabricantes cumplir unos mínimos de calidad (por ejemplo, que en las homologaciones de coches se revise el software y sus

características, etc.)? ¿Serán los seguros más baratos cuanto más fiable y seguro sea el software que va a bordo de los automóviles pues menos se tendrá que ir a repararlo?

5. Conclusiones

La implementación de las características de seguridad (*safety*) y dependabilidad del software en sistemas críticos es exigida a través de estándares en diferentes campos de aplicación e implica la utilización de técnicas de prevención, tolerancia y eliminación de fallos de software – aunque no se utilizan mucho ni con total rigor. Requieren tanto la realización de actividades 'extra' respecto de las mínimas definidas por ejemplo en ISO/IEC 12207 [2], como el uso de técnicas y métodos específicos para asegurar el desarrollo, la verificación, validación y evaluación de la seguridad y dependabilidad, siendo más estrictos cuanto más crítico sea el software. Además, se suele requerir una organización específica, donde ciertos roles (por ejemplo, evaluador de la seguridad, pruebas de validación, etc.) sean más o menos independientes de los desarrolladores. Hay además otros requisitos adicionales a ser cumplidos: los relativos tanto a software que se quiera reutilizar en un producto crítico, como a la cualificación de las herramientas de desarrollo, verificación y validación (pues pueden afectar a la 'calidad' del producto resultante), a los sistemas configurados por datos, etc.

Además de que estos requisitos no se utilizan sistemáticamente, su uso no asegura el 100% de fiabilidad ni de seguridad, son caros de implementar, la duración y complejidad del proyecto son mayores, etc. Hoy, en muchos de estos sistemas, son los fabricantes de estos productos/sistemas críticos los que definen los riesgos que asumirá el usuario final, y los definen en muchas ocasiones en base factores de ventas y mercado y no a los riesgos de seguridad. Por otro lado, los usuarios desconocen estos sistemas tan sofisticados y complejos, difíciles de entender, de mantener y los riesgos que conlleva su utilización. ¿Quién deberá salvaguardar estos mínimos de seguridad y satisfacción? ¿Es posible definir más regulación y legislación al respecto?

Referencias

[1] Rodriguez Dapena, P., *Software Safety Verification in Critical Software*. Tesis doctoral, Universidad Técnica de Eindhoven, Holanda, 2002.

[2] ISO. *ISO/IEC 12207 Systems and software engineering - Software life cycle processes*, ISO, 2008.

[3] IEC. *IEC 61508-3: Seguridad funcional de los sistemas eléctricos electrónicos electrónicos programables relacionados con la seguridad. Parte 3: Requisitos del software (soporte lógico)* Primera edición 1998-12 + Corrig. 1999-04.

[4] RTCA. *DO-178B. Software Considerations in Airborne Systems and Equipment Certification.* RTCA, 1992.

[5] UK MOD. *DEF STAN 00-55(PART 1). "Requirements for Safety Related Software in Defence Equipment", Part 1: Requirements.* UK MOD 1 Agosto 1997.

[6] AENOR. *UNE-EN 50128 Aplicaciones ferroviarias. Sistemas de comunicación, señalización y procesamiento. Software para sistemas de control y protección de ferrocarril.* AENOR, Octubre 2002.

[7] NASA. *NASA-GB-8719.13. NASA Software Safety Guidebook*. Prepared by NASA Safety and Mission Assurance Office. Marzo 31, 2004.

[8] ISO. *ISO/IEC 15026:1998 Information technology -- System and software integrity levels*. ISO/IEC Edition: 1, 1998.

[9] Risks digest. Forum On Risks To The Public In Computers And Related Systems. ACM Committee on Computers and Public Policy. http://catless.ncl.ac.uk/Risks/

Visión innovadora de la calidad del producto software

Antonio Calero, Paco Castro, Hugo Mora, Miguel Ángel Vicedo, David García
Company for Software and Development (CSD)
{acalero, fcastro, hmora, mavicedo, dgarcia}@csd.com.es

Resumen

El desarrollo y la ingeniería del software se han venido complicando día a día en aras de conseguir desarrollos y soluciones cada vez más completas, robustas y que respondan a los cada vez más exigentes requisitos de las organizaciones. El proceso de aseguramiento de calidad tiene como misión principal garantizar todos los requisitos de calidad establecidos, no sólo los que se refieren a un aspecto funcional, sino también a la fiabilidad, eficiencia, portabilidad, mantenimiento,… Para ello, los controles de calidad no deben aplicarse únicamente al código generado, sino que van mucho más allá, y deben recorrer elementos como las librerías de terceros, la arquitectura, la infraestructura tecnológica (servidor de aplicaciones, base de datos, integración), sin olvidar la seguridad, los procesos de construcción y la documentación. Para cada uno de estos elementos definiremos un conjunto de actividades a realizar, una serie de herramientas de soporte y un conjunto de resultados a obtener.

Palabras clave: calidad de software, aseguramiento de calidad, modelo de calidad, buenas prácticas, control de calidad.

Innovative vision of software product quality

Abstract

The development and engineering of software is becoming more complicated by the day, as we strive to achieve increasingly comprehensive and robust solutions which respond to companies' exacting requirements. The primary objective of the quality assurance process is to guarantee compliance with all of the established quality requirements, not only those which are related to functionality but also those that apply to reliability, efficiency, portability and maintainability. To attain this goal quality controls should not only be applied to source code but also must go far beyond, to encompass areas such as third-party libraries, architecture, infrastructure (application server, database server, integration environments), security, build, packaging and documentation. For each of these areas we define a set of quality assurance activities to be executed, a set of support tools and a specification of the results that should be obtained.

Key words: software quality, quality assurance, quality model, best practises, quality control.

Calero, A., Castro,F., Mora, H., Vicedo, M.A. y García, D., "Visión innovadora de la calidad del producto software", REICIS, vol. 5, no.2, 2009, pp.49-55. Recibido: 22-6-2009; revisado: 6-7-2009; aceptado: 31-7-2009

1. Introducción

El software es cada día mucho más complejo, a todos los niveles. A nivel funcional, cada vez le pedimos que haga más cosas, y más importantes, y a nivel técnico, la evolución de las tecnologías implican la adaptación de nuestro software en todo momento. Pero no sólo se trata de complejidad sino también de criticidad. El software es cada vez más crítico, automatizando aspectos más importantes del negocio, y eso tiene como consecuencia que cualquier error o problema que ocasione el software más impacto tendrá sobre las organizaciones. La tendencia actual en las organizaciones es la centralización del software y el empleo cada vez mayor de aplicaciones con arquitectura web, esta característica que permite usar las aplicaciones en cualquier momento y lugar obliga casi siempre a contemplar aspectos como la alta disponibilidad, tolerancia a fallos y el rendimiento. Esto requiere un control absoluto de todos los procesos implicados y además a todos los niveles, ya que cualquier interacción con el sistema podría desestabilizarlo. Lo habitual es que el proveedor de software garantice la calidad de su desarrollo, sin embargo esto no es suficiente, ya que hay que garantizar el correcto funcionamiento de todas las soluciones implantadas (infraestructura, comunicaciones, puestos, etc.).

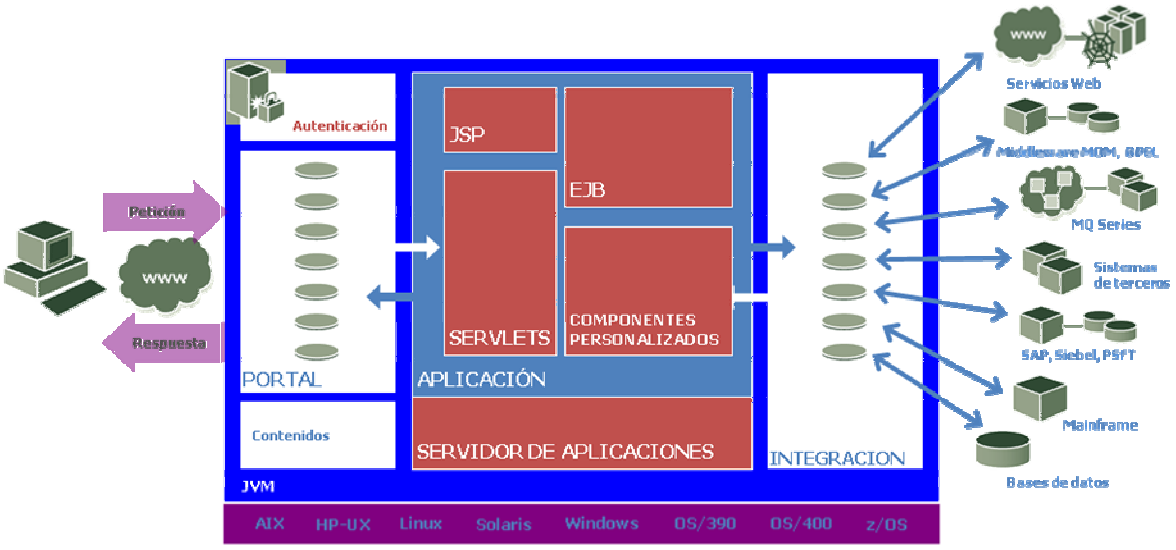

Figura 1. Elementos que participan en la puesta en marcha de los sistemas de información actuales.

2. Puntos de fallo potenciales

Entre los elementos clave de un sistema de información [1] consideramos como puntos de fallo potenciales los siguientes elementos:

- Balanceador de carga de las peticiones entre los servidores
- Servidores web de la capa frontal
- Servidores de autenticación y validación.
- Servidores de aplicaciones de la capa de negocio
- Servidores de base de datos
- Entornos de red y comunicaciones
- Servidores de backup y recuperación
- Cualquier otro sistema de información externo con el que exista una dependencia

A continuación reflejamos algunos ejemplos de puntos de fallo críticos.

2.1. Servidores Web

Los servidores web son un elemento fundamental en una arquitectura centralizada ya que son el punto de entrada a nuestro sistema. En prácticamente la totalidad de los casos el síntoma de la aparición de un problema es la pérdida de rendimiento. Las causas pueden venir por: consumo excesivo de recursos (CPU, memoria), servidores de aplicaciones o sistemas externos no disponibles, acceso a ficheros estáticos muy grandes y pesados, sistemas de ficheros que alcanzan el límite de tamaño, peticiones repetidas o "ecos" (cuando el usuario pulsa varias veces el mismo botón porque no responde el sistema).

La solución en la mayoría de organizaciones pasa por la ampliación de los recursos hardware, sin embargo, aunque parezca la solución sólo lo será puntualmente, ya que la clave es optimizar el rendimiento de las aplicaciones [2] y aplicar los parches correspondientes en el software de base.

2.2. Servidores de aplicaciones

Los servidores de aplicaciones son el núcleo de ejecución fundamental de nuestros sistemas de información por lo que cualquier problema que surja afectará de manera significativa a todos los componentes de nuestro sistema. Al igual que con el resto de componentes el síntoma principal es la pérdida de rendimiento, y viene ocasionado por: problemas de memoria de las aplicaciones y liberación de recursos, mala gestión de los drivers de acceso a base de datos, de la sesión de usuario y del pool de conexiones, acceso a disco elevado, tratamiento erróneo de cadenas de caracteres, transacciones largas e innecesarias.

Una vez más la solución habitual es ampliar los recursos hardware. Esto mejorará el rendimiento puntualmente pero tarde o temprano los problemas volverán a aparecer. De nuevo la clave es optimizar los sistemas y mantener el software de base actualizados.

2.3. Servidores de base de datos

Las bases de datos son otro elemento fundamental en nuestro sistema de información, y también suelen experimentar la mayor parte de nuestros problemas. Algunas de las posibles causas de problemas son: consultas pesadas, interbloqueos, descontrol en el número de sesiones abiertas, acceso a disco, falta de índices, elevado número de cursores abiertos.

De nuevo el síntoma experimentado en el sistema es la pérdida de rendimiento, y de nuevo la solución que se suele tomar es la ampliación de recursos hardware. En su lugar, la experiencia nos dice que en la mayoría de veces basta con optimizar los desarrollos y mantener al día el software de base de datos.

3. Visión y Modelo de Calidad de Software

Llegados a este punto nos damos cuenta de que problemas supuestamente resueltos no lo están, y que tenemos arquitecturas muy complejas en las que detectar el problema es muy difícil, y se hace necesario y fundamental establecer una monitorización continua [3] y la ejecución de actividades de aseguramiento de la calidad.

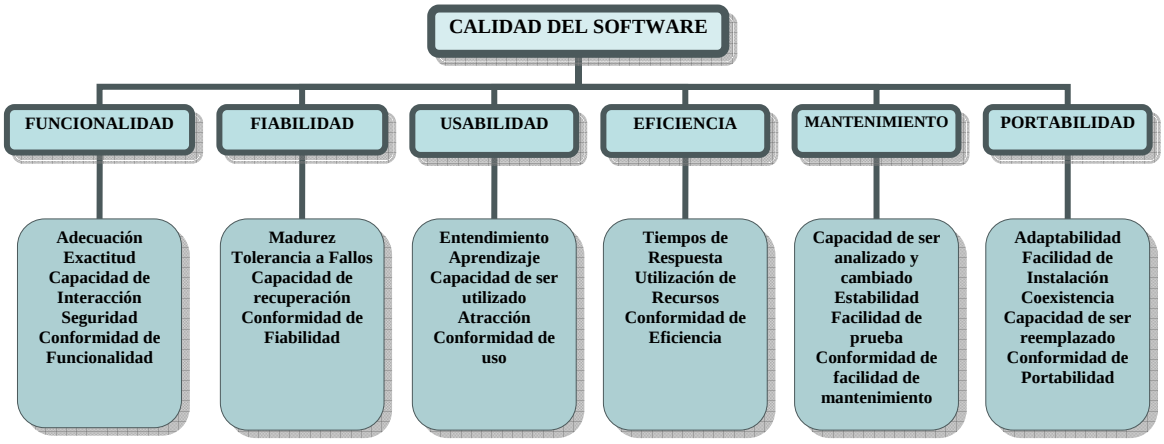

Figura 2. Propiedades de la calidad del software según ISO 9126

El término calidad se define como “propiedad o conjunto de propiedades inherentes a una cosa, que permiten juzgar su valor”. Por tanto, la calidad del software estará definida por el conjunto de propiedades del software que nos permitirán conocer si un software es

mejor o peor que otro, y concretamente estas propiedades vienen definidas por la norma ISO 9126 (estándar internacional para la evaluación de la calidad del software [4]).

El aseguramiento de la calidad del software conformará el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto cumple con los requisitos de calidad establecidos. El proceso de aseguramiento de calidad debe garantizar no sólo los aspectos que se refieren a cuestiones funcionales, sino también los que se refieren al resto de propiedades definidas (eficiencia, usabilidad, fiabilidad, portabilidad, etc.).

El modelo de calidad presentado establece que los elementos a revisar van desde el núcleo con el código fuente hasta la documentación pasando por la infraestructura de ejecución o los sistemas dependientes externos.

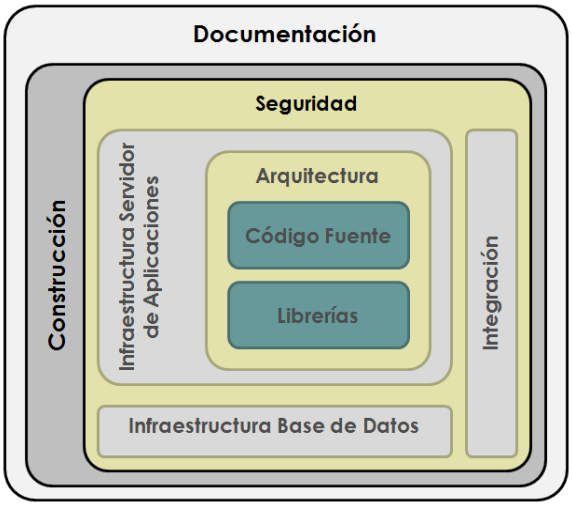

Figura 3. Visión conceptual del modelo de calidad de software

A continuación resumimos algunas de las revisiones que forman parte de nuestro modelo de aseguramiento de la calidad:

- Código fuente. Revisiones de convenciones de código, de pruebas unitarias y cobertura, trazas, control de versiones, complejidad, etc.
- Librerías. Revisiones de licencias y versiones de librerías utilizadas, dependencias y solapamientos, uso de estándares, etc.
- Arquitectura. Revisiones del modelo de arquitectura, consistencia, escalabilidad, mantenibilidad y rendimiento, etc.

- Infraestructura. Revisión de cachés y dependencias, revisiones de configuración en los servidores de aplicaciones y servidores web, parametrización, ficheros de trazas, etc.
- Base de datos. Uso de estándares, revisión de índices, particionamiento, históricos, procesos en segundo plano, controles de versiones del modelo, etc.
- Seguridad. Seguridad en los puestos, seguridad en las integraciones y comunicaciones, planes de contingencia y continuidad, etc.
- Construcción. Revisión de los modelos de construcción de los desarrollos (integraciones, migraciones, etc.), automatización de reglas, empaquetado y ejecución, etc.
- Documentación. Creación de sitios web para los desarrollos, automatización y revisión de la documentación, etc.

Un aspecto muy importante a tener en cuenta en todos los procesos de desarrollo de los productos es la integración continua. La integración continua tiene como finalidad prevenir errores muy comunes, adelantando todo aquello que pueda ser automatizado en el proceso de construcción de un producto. El objetivo es que con una frecuencia determinada se lleven a cabo esas tareas y en caso de detectar errores se proceda a su resolución cuanto antes, evitando que los errores aparezcan el día de publicación de una nueva versión. La integración continua se basa en un motor de reglas que permite automatizar la ejecución de todas esas tareas comunes y contribuye enormemente a mejorar la calidad del software.

3. Conclusiones

El aseguramiento de la calidad del software es una actividad continua durante todo el ciclo de vida de un sistema de información. El ámbito es global, desde el código fuente hasta la infraestructura, seguridad y documentación, por lo que requiere un equipo técnico multidisciplinar y herramientas de soporte específicas.

No sólo hemos de garantizar que el código fuente es de calidad, sino también como este código enlaza con las librerías de terceros, con la base de datos, con los servicios de integración, etc. etc. Sólo así podremos tener la confianza en que todos los requisitos de calidad se están cumpliendo.

Referencias

[1] Casanova, J.C., Calero, A., Devesa, J., "Arquitectura de Sistemas y Aplicaciones en Entornos Críticos: La experiencia de la Conselleria de Sanidad", *IX Congreso Nacional de Informática de la Salud, INFORSALUD 2006, Madrid*.

[2] Calero, A., Gómez, J., "Gestión del Rendimiento y Calidad en Aplicaciones J2EE", *X Congreso Nacional de Informática de la Salud, INFORSALUD 2007, Madrid*.

[3] Calero, A., Casanova, J.C., Devesa, J., "Entornos de Producción Críticos en los Sistemas de Información Sanitarios. Monitorización. Diagnóstico", *IX Congreso Nacional de Informática de la Salud, INFORSALUD 2006, Madrid*.

[4] ISO (International Organization for Standardization), Software Product Evaluation. Quality Characteristics and Guidelines for their Use. ISO/IEC Std 9126, ISO, 2001.

El análisis de anomalías detectadas en las pruebas de software: una vía para mejorar el ciclo de vida

Ramón Enrique González Peralta
Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2½, Boyeros, Ciudad de La Habana, Cuba.
regonzalez@uci.cu

Resumen

Este trabajo incluye un aspecto de mejora continua al proceso de prueba a los requisitos, que consiste en la aplicación de un procedimiento para el análisis y clasificación de las anomalías detectadas, según la naturaleza del error, para identificar los principales problemas cometidos por los desarrolladores e identificar medidas correctivas, contribuyendo, como lo muestran los resultados de esta investigación, a "mejorar" el ciclo de vida.

Palabras clave: prueba de los requisitos, análisis y clasificación de anomalías, medidas correctivas

Analysis of anomalies detected during software testing: a way to improve life cycle

Abstract

This article includes a continuous improvement process to test the requirements, which consists of applying a procedure for analysis and classification of anomalies, according to the nature of the error, to identify major problem committed by the developers and to identify corrective actions, contributing, as shown by the results of this investigation, to "improve" the life cycle.

Key words: test the requirements, analysis and classification of anomalies

González-Peralta, R.E., "El análisis de anomalías detectadas en las pruebas de software: una vía para mejorar el ciclo de vida", REICIS, vol. 5, no.2, 2009, pp.56-62. Recibido: 22-6-2009; revisado: 6-7-2009; aceptado: 31-7-2009

1. Introducción

Para asegurar alta calidad en proyectos de gran tamaño hay que llevar a cabo diferentes actividades, que se pueden desglosar en dos grandes tipos: la verificación y la validación, que abarcan una amplia lista de actividades de Aseguramiento de la Calidad de Software (SQA) que incluye: revisiones técnicas formales, auditorías de calidad y de configuración, monitorización de rendimientos, simulación, estudios de factibilidad, revisión de la documentación, revisión de la base de datos, análisis algorítmico, pruebas de desarrollo,

pruebas de validación y pruebas de instalación [1]. En cada una de estas actividades se detectan anomalías. Cualquier condición que implique la desviación de la especificación de los requisitos, de los documentos del diseño, documentos de usuario, estándares etc. o de la perspectiva y/o experiencia de alguien. Las anomalías pueden ser encontradas (aunque no únicamente) durante las revisiones, pruebas, análisis, compilación o el uso de la documentación de una aplicación [3], que son luego corregidas por el equipo de desarrollo y por lo general los procesos de verificación o validación terminan allí.

El autor de este artículo se plantea la siguiente interrogante: ¿Cómo lograr una retroalimentación constante del proceso de pruebas que contribuya al perfeccionamiento de los productos de trabajo?

2. Materiales y métodos

Para desarrollar la investigación que sustenta este artículo fueron utilizados dentro de los métodos teóricos el **Hipotético-Deductivo** y la **Modelación**. Además para el desarrollo de esta investigación, resultó muy provechoso el empleo de los métodos empíricos, entre ellos: La Observación Participante, la entrevista y el experimento: La Figura 1 muestra visualmente una descripción de los pasos realizados en el experimento, donde: 1ro: Se tiene un personal cuyo producto de trabajo no posee buena calidad; 2do: A través de un Proceso de Pruebas (Bien definido) se detectan las anomalías por cada producto de trabajo de cada desarrollador; 3ro: Luego de identificadas las tendencias se ejecutan las medidas de preparación del personal; 4to: se realiza nuevamente otro proceso de pruebas a los mismos desarrolladores y se recopilan de nuevo los datos; 5to: por último se comprueba el estado del personal y se comparan las cantidades de anomalías, para comprobar la hipótesis.

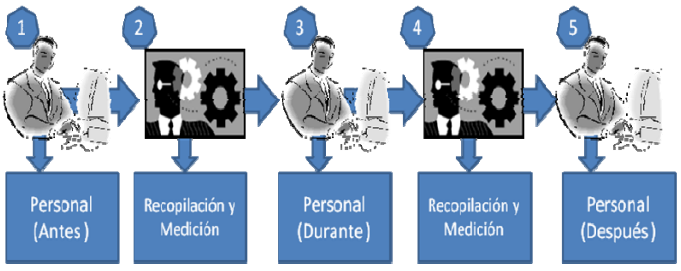

Figura. 1 Pasos seguidos en el experimento

Existen muchas formas de identificar las desviaciones de las especificaciones e incumplimientos con los planes durante el proceso de desarrollo, el uso indiscriminado de

dichos conceptos puede traer confusión dentro del proceso, por lo que en esta investigación se asumió el concepto dado en la IEEE std 1044 "Standard Classifcation for Software Anomalies" [3], mencionado en la introducción. Este concepto se asume por la necesidad de estandarizar la forma de llamar a las desviaciones que se detectan durante el proceso de prueba de los requisitos, se puede afirmar que el estándar [3] contiene una metodología basada en un proceso que va desde el reconocimiento de la anomalía a la aplicación de medidas correctivas, este proceso contiene varios pasos que se describen en detalle en la norma. El estándar no es una camisa de fuerza, el mismo permite al usuario la adaptación de su procedimiento y de las clasificaciones para que se sea aplicado en un proyecto atendiendo a las características específicas del mismo.

En la investigación base de este artículo se realizó el estudio del estándar IEEE std 1044, a partir del cual se asumieron algunas de las clasificaciones que este muestra y se adaptó su proceso para la clasificación de las anomalías. De forma general el Proceso de Clasificación de Anomalías, quedaría como se muestra en la siguiente figura (Fig. 2).

Figura 2. Proceso de Clasificación de Anomalías

Este proceso está compuesto por una secuencia de pasos, mediante los cuales se detectan, registran y clasifican las anomalías, y además se identifican las medidas correctivas que deben aplicarse. Los responsables de la realización de estas actividades se muestran entre paréntesis al final del nombre de cada una de ellas.

2.1. Clasificaciones empleadas

A continuación se exponen las clasificaciones fundamentales para las anomalías que se detectaron en el desarrollo del Proceso de Pruebas, separadas en las clasificaciones a la documentación y en las clasificaciones en las aplicaciones informáticas. Es necesario aclarar que las clasificaciones que propone el estándar IEEE std 1044 [1] no son exactamente las que se asumen en el desarrollo de TODOS los tipos y tamaños de software,

estas clasificaciones deben adaptarse a las características específicas del proceso en cuestión. A continuación se muestra un caso particular en el que se desarrolló la investigación, para el desarrollo del SIIPOL v1.0 (Sistema de Investigación e Información Policial) en la Universidad de las Ciencias Informáticas (UCI), en el presente trabajo sólo se exponen las clasificaciones para las aplicaciones de software; no obstante las clasificaciones para la documentación existen y también se aplican.

Clasificación	**Identificador**
Validación	AVA
No correspondencia con la Especificación	NCEA
Clasificación	Identificador
Ortográfico(en textos de las interfaces del software)	AOA
Seguridad	ASA
Rendimiento	ARA
Funcionalidades “visibles” no implementadas	FVNA
Funcionales	RFA
Relacionadas con la navegación	RNA
Relacionadas con los reportes	RRA
Otras Anomalías	OAA

Tabla 1. Clasificaciones de las anomalías detectadas en aplicaciones informáticas

No	**Clasificación para las acciones correctivas**
1	A nivel de departamento
1.1	Revisar el proceso (políticas/procedimientos)
1.2	Capacitar al personal
1.3	Crear/Revisar/Reforzar el uso de estándares y especificaciones
1.4	Reubicar personas y/o recursos
1.5	Mejorar y/o reforzar las auditorías
2	En instituciones educativas y de investigación
2.1	Investigar el problema
2.2	Desarrollar nuevas tecnologías
2.3	Identificar enfoques alternativos de prueba
2.4	Crear y/o revisar las pruebas
2.5	Reforzar los estándares educacionales

Tabla 2. Clasificaciones para las acciones correctivas

2.2. Medidas correctoras a aplicar

Luego de clasificadas las anomalías y de verificada su posible causa, es necesario proponer medidas correctivas para solucionar las mismas, con tal propósito, se definieron los diferentes niveles a los que pueden ser aplicadas las medidas correctivas propuestas y algunas de las clasificaciones para las mismas, todo esto a partir de lo planteado en el estándar IEEE std 1044 [3]. Las medidas se muestran en la Tabla 2.

3. Resultados y discusión

Para la aplicación de la solución se seleccionó un grupo de 7 módulos, para los cuales se especifica en la tabla 3 la cantidad de Casos de Uso (CU) que lo componen así como la cantidad de Diseñadores-Programadores que implementaron esos CU.

No	Módulo	Cantidad de CU	Cantidad de Desarrolladores
1	Gestión Administrativa	9	2
2	Investigación Penal	35	4
3	Investigación Criminalística	22	5
4	Investigación Forense	1	1
5	Análisis de Información	4	3
6	Estadísticas	1	1
7	Registro y Control	1	1
8	Total	73	17

Tabla 3. Muestra seleccionada

Durante la realización del experimento en la investigación base de este artículo se realizaron dos ciclos completos de Pruebas. A continuación se muestran los resultados obtenidos en los ciclos de prueba y el análisis de los mismos:

Paso 1: Se seleccionó la muestra a ser empleada para la aplicación de la solución (personal y sus productos de trabajo), como resultado de este paso se obtuvo la información que se muestra en la tabla 3, para los desarrolladores, recibiendo como información de entrada el hecho de probar una versión del software con 73 CU implementados por 17 programadores.

Paso 2: Luego de realizado el primer ciclo de pruebas se detectaron y analizaron en el CCC las anomalías detectadas quedando de la siguiente forma la relación entre

las detectadas y las Reales (las que quedaron luego de separar las rechazadas por el comité de control de cambios). Al clasificar las Anomalías por tipo de error, quedaron de la siguiente forma, separadas por módulos (las abreviaturas utilizadas para los tipos de anomalías se corresponden con los identificadores de estas clasificaciones listados en la tabla 1). Se detectaron 548 anomalías reales, quedando un promedio de Anomalía por CU de: 7.5, cuestión que no es favorable, teniendo en cuenta que la tasa deseada de errores para asegurar calidad de SW en este caso debe ser menor de una anomalía significativa por CU. Los principales problemas detectados fueron: Anomalías en las funcionalidades del sistema (120), errores ortográficos (98), desviaciones de la ECU (73), anomalías en las validaciones (71) funcionalidades visibles no implementadas (62).

Paso 3: Identificación y aplicación de las medidas correctivas. Las acciones correctivas que se listan a continuación siguen las clasificaciones definidas en el la tabla 2:

- A nivel de equipo (Departamento): revisar el proceso (políticas/procedimientos) y reubicar personas y/o recursos.
- A nivel de institución docente: identificar enfoques alternativos de prueba y crear y/o revisar las pruebas.

Paso 4: Ejecución del segundo ciclo de pruebas.

Luego de haber ejecutado las medidas correctivas sobre el personal se seleccionaron otros productos de su trabajo con características similares a la muestra anterior, ya que en muchos casos hubo artefactos que tuvieron que cambiarse completamente.

Paso 5: Luego de realizado el segundo ciclo de pruebas se realizó una comparación entre ambos ciclos y se pudo corroborar la efectividad de la solución, pues el personal sobre el cual se ejecutaron las medidas correctivas mejoró considerablemente en la producción de sus siguientes artefactos. La figura 3 incluye una gráfica que demuestra esta afirmación.

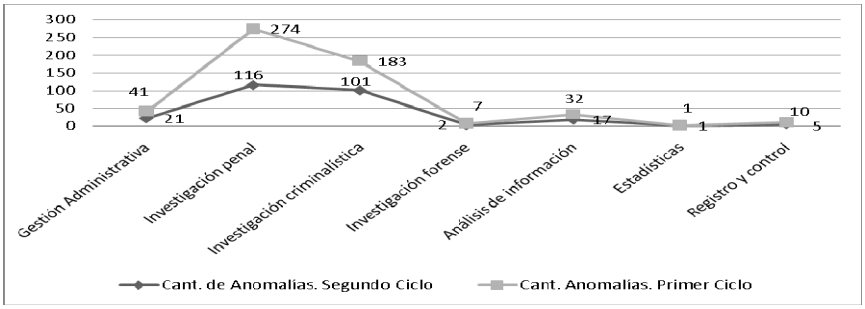

Figura 3. Comparación entre ambos ciclos de prueba

Es evidente entonces que añadiendo un procedimiento al proceso de pruebas que incluya la clasificación de las anomalías detectadas y la identificación de medidas correctivas se mejorará en gran medida el ciclo de vida del software.

4. Conclusiones

Con el presente artículo se ha propuesto, aplicado y comprobado un procedimiento para la clasificación de anomalías. Queda demostrado que debe seguirse perfeccionando este aspecto, ampliando la cantidad de medidas correctivas a aplicar y utilizando métodos más eficientes en la selección de la muestra, además se debe implementar este procedimiento en otros proyectos de desarrollo de software.

Referencias

[1] Pressman, R.S., *Ingeniería del Software. Un enfoque práctico,* Mc Graw Hill, 2002.

[2] Boehm B.W., *Software Engineering Economics*, Prentice-Hall, 1981.

[3] IEEE, *IEEE Std. 1044 Standard Classification for Software Anomalies,* IEEE, 1993.

[4] CHÁVEZ, I. R. S. Verificación y Validación del Software. Revisiones de SW (VyV Estática), Monterrey, 2005.

Experiencias de una PYME en la mejora de procesos de pruebas

Antonio de Rojas
Clave Informática, S.L.
Galileo Galilei, 12 - Elche Parque Industrial – España
aderojas@clavei.es
Tanja E.J. Vos, Beatriz Marín
Centro de Investigación en Métodos de Producción de Software (ProS)
Universidad Politécnica de Valencia - España
{tvos, bmarin}@pros.upv.es

Resumen

Las metodologías más habituales para la mejora de procesos de pruebas están orientadas a organizaciones grandes y son, por tanto, difícilmente aplicables al perfil de las PYME, con recursos limitados y poca madurez en estos procesos. En un trabajo anterior, se han propuesto una serie de acciones sencillas y concretas que las PYME pueden realizar para mejorar los procesos de testeo; sin dedicar muchos recursos, obteniendo resultados rápidamente y preparándolas para un posterior proceso de mejora más formal. En este trabajo, se presentan las experiencias que una PYME ha obtenido luego de implantar algunas de las acciones propuestas.

Palabras clave: Pruebas, Software, PYME, Mejora de Procesos.

SME experiences in testing process improvement

Abstract

Common methodologies for test process improvement are oriented towards big companies, and so they are more difficult to apply to SMEs that have limited resources and immature processes. In a previous work, a set of concrete and simple actions were proposed in order to SMEs can improve their test processes without dedicating too many resources. The underlying idea is that they will quickly obtain results that will prepare them for a more formal improvement process. These results will come from real needs of the specific companies and not from artificial needs listed in process improvement books. In this paper, experiences from an SME following recommendations of previous work will be presented,

Key words: Testing, Software, PYME, Process Improvement.

De Rojas, A., Vos, T. y Marin, B., " Experiencias de una PYME en la mejora de procesos de pruebas", REICIS, vol. 5, no.2, 2009, pp.63-69. Recibido: 22-6-2009; revisado: 6-7-2009; aceptado: 31-7-2009

1. Introducción

Existen bastantes metodologías y técnicas para la mejora de procesos de pruebas, como por ejemplo Test Improvement Model (TIM) [2], Testing Organization Maturity Model

(TOM)[2], Testing Madurity Model (TMM) [1], o Test Process Improvement (TPI) [3]. Estas metodologías han sido diseñadas para organizaciones más grandes, y no son fácilmente adaptables a estructuras mucho más pequeñas, como las PYME. El principal problema es que el coste y la duración de un proceso de mejora son desproporcionados respecto a los recursos disponibles en una PYME.

Este artículo describe los resultados en la mejora de los procesos de pruebas que ha obtenido la empresa Clave Informática, una PYME de la Comunidad Valenciana (España) que se dedica a diseñar y comercializar aplicaciones software de Producción, Gestión Comercial y Financiera, todas ellas integradas con Servicios ISP/IDC (Internet Service Provider / Internet Data Center) de alta disponibilidad y seguridad. Las acciones de mejora en Clave Informática se han basado en un trabajo anterior de uno de los autores de este artículo [4], donde se propone una serie de acciones sencillas y concretas que las PYME pueden realizar sin dedicar muchos recursos, obteniendo resultados rápidamente y preparándolas para un posterior proceso de mejora más formal.

El resto del artículo está organizado como sigue: la Sección 2 describe cómo preparar a una PYME para la mejora de procesos de pruebas, la Sección 3 presenta las experiencias de Clave Informática, y la Sección 4 presenta algunas conclusiones.

2. Preparando a las PYME para la Mejora de Procesos de Pruebas

Antes de aplicar alguna metodología para la mejora de los procesos de pruebas en una PYME, es necesario primero aplicar acciones más sencillas para preparar a las PYME para mejorar estos procesos. En [4], se proponen 3 acciones muy prácticas y sencillas:

1. Formación y concienciación de la importancia de las pruebas del software.
2. Los programadores deben realizar algún tipo de pruebas unitarias sobre sus desarrollos, dejando a su elección qué y cómo hacerlo. Es necesario confiar en el conocimiento de los programadores: ellos saben lo que quieren programar, por lo tanto, que sean ellos los que comprueben si lo hacen correctamente.
3. Asignar (o contratar) al menos 1 persona dedicada exclusivamente a realizar pruebas de alto nivel, es decir, a realizar pruebas de aceptación e integración.

[2] Gerrald Consulting web site, http://www.gerrardconsulting.com/default.asp?page=/tomoverview.html

Para empezar, en [4] no se define un proceso estructurado de pruebas, simplemente se debe probar de la mejor forma posible, con el objetivo de encontrar errores y con la libertad para realizar las pruebas según se crea conveniente. Así, además de encontrar errores, se sacarán a la luz necesidades para realizar más eficazmente el trabajo (p. ej.: definición de requisitos, gestión de defectos, análisis de riesgos, planificación, etc.). Estas necesidades se transformarán en acciones que poco a poco mejoran los procesos de pruebas hasta llegar a procesos estructurados, sin depender de un modelo formal y la implantación de acciones artificiales que éste dicte.

3. Las Experiencias de Clave Informática

La producción de software de Clave Informática gira alrededor de dos ERPs, que son aplicaciones cliente-servidor diseñadas en arquitectura de 3-capas. Los ERP han sido desarrollados con tecnología Microsoft y utilizan SQL Server como base de datos.

A partir de finales de 2003, Clave adoptó una serie de medidas en el proceso de desarrollo de software (vea la tabla 1). Estas medidas se tradujeron en instrucciones que dieron consistencia al proceso de desarrollo de software y entendimiento a todos los colaboradores. Luego, se establecieron los primeros indicadores para evaluar el proceso: Incidencias por Versión, Incidencias por Módulo, e Incidencias por Aplicación. De esta manera, Clave introdujo procesos de pruebas en la fase de desarrollo, pero a nivel de detalle todavía se carecía de procesos estructurados que permitiesen mejorar las pruebas del software y resolver dudas en cuanto a: (1) ¿Cómo empezar a probar las aplicaciones que ya tenían desarrolladas (con miles y miles de líneas de código)? y (2) ¿Cómo especificar la forma en que se debían probar los nuevos desarrollos? Además, el analista (o incluso el jefe de producto) ya no era la persona idónea para realizar las pruebas cuando el objetivo era disminuir el número de incidencias en las versiones finales.

Hacia finales de 2006, estos inconvenientes impulsaron a Clave Informática a establecer un "Comité de Calidad del Software". Hablando con los autores del artículo [4], el comité decidió contar con personal exclusivo para la realización de las pruebas de alto nivel (aceptación e integración), y buscar ayuda con la formación y concienciación de la importancia de las pruebas del software. Gracias a convenios entre Universidad y Empresa, dos estudiantes de Ingeniería Técnica en Informática de la Universidad de Alicante

realizaron sus prácticas en Clave. Ellos tenían como cometido la puesta en marcha de procesos de pruebas bajo la tutela de la Universidad Politécnica de Valencia. Al final del período, Clave decidió contratar a tiempo completo a uno de los estudiantes.

Descripción Medida
1. Procedimientos que refuercen las pruebas unitarias por parte del propio programador mediante la creación de una plantilla de chequeos básicos por tipo de aplicación, que le sirvieran como puntos de verificación de cuestiones básicas a contemplar en el desarrollo. Por ejemplo, chequeos de la declaración de variables, chequeos de los colores de la interfaz, chequeos de nombres de procedimientos, etc.
2. Definición de un flujo de trabajo hacia el analista del sistema, que fue asignado como el encargado de realizar la verificación y validación de las modificaciones realizadas por un programador. La verificación se basaba en pruebas exploratorias a alto nivel con el objetivo de detectar errores en el desarrollo, los cuales generaban otro flujo de trabajo de vuelta al programador para su resolución, y así sucesivamente.
3. Desarrollo de módulos personalizados dentro de los ERPs, para incorporarles herramientas que facilitasen la gestión de todos los procesos. Se trataba de la implementación de un módulo de Gestión de proyectos y un módulo de Incidencias de Software para la captura y resolución de errores de programa.

Tabla 1. Medidas adoptadas por Clave Informática para conseguir la certificación ISO 9001

Las pruebas de aceptación resultaron un gran avance en Clave, puesto que están muy enfocadas tanto a la verificación y la validación de modificaciones y adaptaciones personalizadas del software a sus clientes. Consecuentemente, durante 2007 se prepararon procedimientos más estructurados de pruebas. Estos procedimientos fueron diseñados en base a las experiencias y necesidades que el testeador a tiempo completo había averiguado durante sus actividades de pruebas. Las pruebas de aceptación se integraron en del ciclo de desarrollo en dos fases: (1) Una vez que el programador finalizaba su desarrollo de manera de verificar naturalmente las modificaciones (así el flujo de trabajo iba del programador hacia el testeador), y (2) Luego de la puesta en marcha del sistema, se decide realizar las pruebas de aceptación al comienzo del proceso de desarrollo (antes de que el programador finalice su trabajo). Esto presenta muchas ventajas, como por ejemplo:

- Permite diseñar las pruebas en total colaboración con el cliente. Al mismo tiempo que se conocen los cambios pedidos, se va diseñando la forma de probarlos.
- Las pruebas no están viciadas por el desarrollo ya realizado por el programador, que puede influenciar la forma de realizar las pruebas.
- El programador puede verificar sus desarrollos siguiendo el diseño de las pruebas de aceptación ya realizado. Así, al testeador llega el trabajo mucho más depurado.

En último lugar, para que las pruebas de aceptación sean realmente efectivas, se ha detectado que es necesario poner en marcha un sistema de integración continua, que permita tener una versión "Beta" disponible todos los días, sobre la cual se realizasen las pruebas en las mismas condiciones que si se tratase de la versión final instalada en los clientes. Siempre existe la posibilidad de realizarlas sobre el código fuente, pero en Clave son conscientes de que el comportamiento puede variar. Finalmente, una vez que los programadores terminan su trabajo, deben actualizar el controlador de versiones. De esta manera, en Clave se ha ideado un proceso de integración propio, que se ejecuta por la noche y que permite la generación de la versión "Beta" para verificar y también la actualización de la base de datos dependiente de esos cambios. Así, el testeador tiene cada mañana una versión totalmente funcional para realizar las pruebas de aceptación.

Además, se introdujo una nueva métrica: el Índice de Errores no Detectados (EnD) para proporcionar una idea de la efectividad de las pruebas. La fórmula usada es:

$$\textbf{Índice Errores No Detectados} = \frac{\text{Errores Detectados por Cliente}}{\text{Errores Detectados por Cliente + Errores Detectados por el Proceso de Pruebas}}$$

Este indicador no sólo contribuye en la evaluación de la efectividad de las pruebas actuales, sino que también ayuda a establecer objetivos de mejora para el futuro. Los resultados de la aplicación de la métrica EnD fueron concluyentes: existe una gran diferencia entre las aplicaciones sometidas a los procesos de pruebas, de las que se habían dejado fuera deliberadamente porque su funcionamiento no era tan crítico. Así, las percepciones internas de la empresa y de los clientes se trasladaron a datos objetivos.

La gráfica en Figura 1, muestra la evolución de todo lo comentado hasta este punto, reflejando el número de incidencias detectadas en los procesos de pruebas desde 2005 hasta 2008. Estas incidencias no llegaron a manos de los clientes debido a que se solucionaron antes de las versiones finales. En esta grafica se comparan los resultados de dos aplicaciones: una sometida a los procesos de pruebas, y otra que se decidió dejar fuera. La aplicación probada fue desarrollada por 5 programadores a tiempo completo y tiene alrededor de 800.000 líneas de código, en cambio, la aplicación sin pruebas fue desarrollada por 3 programadores y tiene alrededor de 500.000 líneas de código. El esfuerzo real de pruebas requirió aproximadamente el 30% del tiempo de desarrollo. El 21% de las incidencias detectadas se clasificaron como graves, es decir: (1) Alta

repercusión económica en el cliente, debido a resultados erróneos o por una posible parada de negocio; (2) Alta cantidad de clientes afectados cuando la incidencia está contenida en un módulo de uso común, .El tiempo de resolución para las incidencias graves es de un día, mientras que para las incidencias leves es de tres días. El 10% de las incidencias corresponde tienen un impacto mínimo en la funcionalidad del programa, o su resolución implica grandes cambios en la estructura de la aplicación. Estas incidencias se planifican para su tratamiento en el siguiente trimestre.

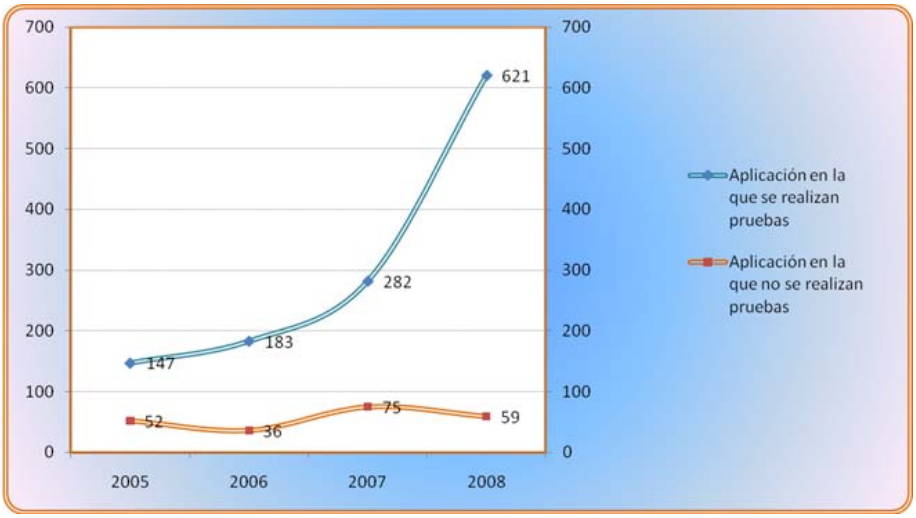

Figura 1: Numero de incidencias detectadas en dos aplicaciones

El análisis de la gráfica en la Figura 1 permite obtener las siguientes conclusiones:

- La introducción de procesos de pruebas mejora los resultados del desarrollo de software (Vea la diferencia de incidencias de ambas aplicaciones en 2005 y 2006).
- Con la incorporación de personal dedicado a las pruebas la mejora es considerable (En 2007 las incidencias son muy superiores a la tendencia de los años anteriores).

La mayor diferencia se produjo entre 2007 y 2008, donde se produjo una mejora exponencial debido a la implantación de las necesidades detectadas por el testeador para realizar eficazmente su trabajo. Por razones de espacio, no se han detallado estas necesidades. Sin embargo, éstas no serían útiles a otras PYME, ya que cada PYME tiene diferentes necesidades dependiendo de su personal, procesos y productos.

4. Conclusiones

En este artículo, hemos descrito el éxito que Clave Informática ha tenido estructurando paulatinamente los procesos de pruebas basándose en las necesidades internas de la empresa detectadas por empezar a probar con el objetivo de encontrar errores. Hoy por hoy, Clave Informática sigue trabajando en la mejora continua, destacando lo siguiente:

- Automatización de Pruebas. Hay procesos que se están realizando de forma manual y repetitivamente, que al automatizarlos se ganaría en productividad y eficiencia.
- Pruebas unitarias. El objetivo es extender como una práctica habitual entre los programadores, la realización de pruebas unitarias para los nuevos desarrollos.

En cuanto al impacto sobre el personal de Clave, las mejoras aquí introducidas han contribuido a: Mejorar la seguridad y bienestar de los desarrolladores, ya que saben que sus resultados están verificados y validados por el testeador; Mejorar la confianza de los consultores y el personal de soporte, ya que saben que los procesos de pruebas están permitiendo liberar software con menos errores; y Mejorar la satisfacción de los clientes, ya que se les está dando lo que necesitan y libre de errores. Por ello, invitamos a otras PYME a leer el artículo [4] y lograr el éxito que ha tenido Clave.

Referencias

[1] Burnstein, I., Suwanassart, T., Carlson, C.R., "The Development of a Testing Maturity Model", En: *Procs of the 9th Int Quality Week Conf, 21-24 de mayo de 1996*.
[2] Ericson, T., Subotic, A., Ursing, S., "TIM-a test improvement model", *Software Testing, Verification and Reliability,* vol. 7, nº 4, pp. 229-246, 1997.
[3] Koomen, T.; Pol, M., *Test Process Improvement: A practical step-by-step guide to structured testing*. Addison-Wesley, 1999.
[4] Vos, T.E.J., Sanchez, J., Mannise, M., "Mejorando el testeo en las PYME ¿Cómo empezar?", En: *Proc de la 5a Ed de las JTS2008, 2-4 de abril de 2008, Valencia.*

Procedimiento para pruebas de intrusión en aplicaciones Web

Delmys Pozo Zulueta, Mairelis Quintero Ríos, Violena Hernández Aguilar, Lisney Gil Loro, Maria Felix Lorenzo Álvarez
Universidad de las Ciencias Informáticas
{dpozo@uci.cu, mquintero, violena, lgil, mflorenzo}@uci.cu

Resumen

Laboratorio Industrial de Pruebas de Software (LIPS) de la empresa cubana Calisoft (Centro para la Excelencia en el Desarrollo de Proyectos Tecnológicos) se encarga de realizar pruebas de liberación a productos entregables de proyectos de exportación e importación. Entre los tipos de prueba que realiza el LIPS están las pruebas de seguridad, específicamente las pruebas de intrusión a aplicaciones Web. La fuerza de trabajo de LIPS son los estudiantes de 1er y 2do año de la UCI (Universidad de las Ciencias Informática). Como estos estudiantes no tienen experiencia y conocimientos en pruebas de seguridad, se decidió elaborar un procedimiento de pruebas de intrusión para aplicaciones Web que guíe a los probadores. Este trabajo presenta las etapas del procedimiento y muestra las pruebas y herramientas de automatización que incluye el procedimiento. También exponen los indicadores de listas de chequeo y plantillas de casos de prueba que se definieron para enseñarle a los probadores que deben probar.

Palabras clave: pruebas de seguridad, procedimiento de pruebas, herramientas de pruebas

Procedure for intrusion testing for Web applications

Abstract

Industrial Testing Laboratory (LIPS) from Cuban Software company Calisoft (Center for the Excellence in Technological Projects's Development) takes upon to accomplish Release Test to produces artefacts of export and importing projects . Among the test types that the LIPS realizes are Security Test, specifically testing of intrusion to Web applications. The students of 1st and 2nd year under grade are LIPS's manpower of the UCI (University of Informatics Sciences). As these students are not experienced and knowledge in Security Test, was decided testers prepare a Intrusion Test Procedure for Web applications to guide to unexperienced tester. This work presents the stages of the procedure and evidences proofs and automatization tools that the procedure includes. Also they expose the indicators of check lists and templates of test cases that were circumscribed to show to the testers what that they should test.

Keywords: security tests, testing procedure, test tools

Pozo D., Quintero M., Hernández V., Gil L., Felix, M. y Álvarez, L.,"Procedimiento para pruebas de intrusión en aplicaciones Web", REICIS, vol. 5, no.2, 2009, pp.70-76. Recibido: 22-6-2009; revisado: 6-7-2009; aceptado: 31-7-2009

1. Introducción

El LIPS constituye uno de los grupos de trabajo de la empresa Calisoft (Centro para la Excelencia en el Desarrollo de Proyectos Tecnológicos). En él se fomenta una correcta instrucción técnica y pedagógica en el desarrollo de habilidades prácticas en los estudiantes a través de las actividades productivas. El LIPS cuenta con un Laboratorio de Pruebas Funcionales con la capacidad de 60 puestos de trabajo a tiempo completo para las pruebas y como fuerza de trabajo parte de los estudiantes de 1ro y 2do año de la UCI (3500 estudiantes aproximadamente). Cuenta además con 21 especialistas de Calidad de Software encargados de dirigir todo el proceso de pruebas en el laboratorio.

En el LIPS se realizan pruebas funcionales, carga y estrés, regresión, exploratorias, volumen y seguridad. Las pruebas de seguridad realizadas son pruebas de intrusión a aplicaciones web debido a la necesidad que existe en la actualidad de evaluar y garantizar la seguridad de estos sistemas. Estudios realizados por CENZIC [1] muestran que en la segunda mitad del 2008 el 80% de las vulnerabilidades encontradas en Internet pertenecían a aplicaciones web. El número de ataques aumenta cada día, ya no basta con garantizar la seguridad durante el proceso de desarrollo, también hay que realizar pruebas de intrusión que evalúen la seguridad del software después de realizado y antes de ser entregado al cliente para su posterior uso.

El proyecto OWASP (*The Open Web Application Security Project*) [2] define la prueba de intrusión o penetración a aplicaciones web como un método de evaluación de la seguridad de un sistema de ordenadores o una red mediante la simulación de un ataque. Una prueba de intrusión está enfocada a evaluar la seguridad de una aplicación web. El objetivo de esta prueba es encontrar las vulnerabilidades de las aplicaciones web y explotarlas para tomar el control del sistema. Como la fuerza de trabajo del LIPS son estudiantes con poco conocimiento y experiencia en pruebas de intrusión surge la necesidad de desarrollar un procedimiento que los guíe en la realización efectiva de las pruebas de intrusión a aplicaciones web.

El presente trabajo muestra las etapas del procedimiento elaborado, además de las pruebas y herramientas incluidas en el mismo. La sección 2 presenta las pruebas y herramientas del procedimiento, la sección 3 muestra sus etapas y la 4 las conclusiones.

2. Pruebas y herramientas que incluye el procedimiento.

La realización de las pruebas se separó en dos niveles siguiendo la filosofía de la compañía Above Security. El primer nivel consiste en la evaluación de las vulnerabilidades de la aplicación, donde se recopilan las debilidades encontradas mediante la utilización de herramientas automatizadas y poca intervención manual, en este nivel las pruebas son realizadas por los estudiantes debido a la poca intervención manual que se requiere. En este nivel es donde se aplica el procedimiento desarrollado.

El segundo nivel consiste en la recopilación de evidencias objetivas que demuestren la explotación de las vulnerabilidades detectadas en el primer nivel. Como este nivel requiere de mayor tiempo, esfuerzo y conocimiento por parte de los probadores para llevar a cabo su ejecución el LIPS contrata los servicios del Grupo de Seguridad (GS) de la UCI, el cual cuenta con probadores expertos en temas de seguridad de sistemas informáticos.

Para la realización del procedimiento se utilizó la Guía de Pruebas del proyecto OWASP. Esta guía define varias categorías de prueba de intrusión, de estas el procedimiento contempla 4. También se seleccionaron dentro de estas categorías las pruebas más fáciles de aplicar por los probadores. Las herramientas automatizadas que se utilizan son el escáner de vulnerabilidades Nessus y el excelente software para extraer parejas de usuario / contraseña de contraseñas remoto Brutus.

Las categorías de prueba del procedimiento son:

- Recopilación de Información: El objetivo de esta categoría es comprobar si la aplicación brinda datos sensibles utilizables por cualquier atacante. Los tipos de prueba incluyen comprobación de firma digital, descubrimiento de aplicaciones, análisis de códigos de error y pruebas de SSL/TLS.
- Comprobación de las reglas del Negocio (CRN): Las pruebas de esta categoría es la comprobación de las reglas del negocio definidas para la aplicación.
- Comprobación de la autenticación (CA): Esta categoría pone a prueba el sistema de autenticación de la aplicación mediante prueba de fuerza bruta y pruebas al recordatorio de contraseñas del sistema.
- Validación de datos (VD): Verifica que todas las entradas de datos estén validadas realizando pruebas de inyección SQL, ORM, LDAP, XML, SSI, procedimientos almacenados y código.

3. Procedimiento de Pruebas de Intrusión

El procedimiento define los pasos para llevar a cabo las pruebas de intrusión desde su planificación hasta la documentación de los hallazgos encontrados, haciendo uso de diferentes plantillas de apoyo.

2.1. Etapa 1: Planificación de las Pruebas

En esta fase el EP planifica las pruebas, para ello utiliza la platilla de Plan de Prueba definida por RUP donde se establecen las especificaciones necesarias para las pruebas, como hardware, recursos del sistema, la estrategia de prueba, cronograma planificado, cronograma real, los roles y responsabilidades.

2.2. Etapa 2: Diseño de las Pruebas

Para guiar a los probadores en la ejecución las pruebas se elaboraron plantillas de listas de chequeo (LCH) y de caso de prueba (CP) que serán rediseñados por el EP en esta etapa según las características de la aplicación. Para probar las categorías de prueba CRN y VD se definieron CP y para las categorías de prueba RI y CA se definieron LCH.

Las LCH recogen los indicadores a evaluar, su descripción, la herramienta que se utilizará, el resultado esperado y el resultado real. Los indicadores a evaluar definidos para la categoría RI de forma general recogen si puede obtenerse la firma digital (tipo y versión) del servidor web, los puertos abiertos en el IP del servidor y los servicios asociados a esos puertos, las aplicaciones web instaladas en el servidor, la existencia cifrados débiles y si el código de error de la aplicación muestra información sobre el sistema operativo o base de datos del servidor web. Los indicadores a evaluar definidos para la categoría CA contemplan si se logra mediante el ataque de fuerza bruta obtenerse el usuario y la contraseña de un usuario privilegiado en la aplicación, si puede saltarse el sistema de autenticación y el del recordatorio contraseña.

Para realizar el CP de la categoría CRN primero se debe realizar una matriz de privilegios que contenga las funcionalidades de la aplicación y los roles que tienen permiso para realizarla. Luego se crea un CP por rol y en él se coloca un escenario para cada funcionalidad donde el rol no tenga permisos, para comprobar si esta funcionalidad puede

ser ejecutada ilegalmente por un rol sin privilegios, o con privilegios mínimos. La estructura del CP se muestra en la tabla 1.

Nombre del Rol	**Escenarios**	**Descripción de la funcionalidad a probar**	**URL**	**Resultado esperado**	**Respuesta del sistema**
< Nombre del Rol>	<Funcionalidad a probar>	<Descripción de la funcionalidad.>	<URL de acceso a la funcionalidad>	< Resultado que se espera al realizar la prueba.>	<Resultado que se obtiene al realizar la prueba.>

Tabla 1. Estructura del CP de la categoría CRN

La tabla 2 muestra el CP elaborado para la categoría VD. Se diseñará un CP por cada funcionalidad reflejando todos los campos de entrada que tiene.

Funcionalidad:	<Nombre de la funcionalidad>		**Herramienta:**	<Nessus>
Campos de Entrada	**Clasificación**	**Resultado Esperado**	**Respuesta del Sistema**	**Flujo Central**
<Nombre del campo de entrada>	<La clasificación es según el componente de diseño utilizado][ejemplo: campo de texto, lista desplegable o campo de selección.>	<Resultado que se espera al realizar la prueba, específicamente si el campo de entrada es vulnerable a inyección SQL, ORM, LDAP, XML, SSI, código o procedimientos almacenados.>	<Resultado que se obtiene al realizar la prueba>	<Pasos a desarrollar para probar la Funcionalidad que se indicó>

Tabla 2. Estructura del CP de la categoría VD

2.3. Etapa 3. Ejecución de la pruebas

En esta etapa los probadores ejecutan las categorías de pruebas definidas haciendo uso de las LCH y los CP diseñado por el EP. La ejecución de las pruebas se realiza comenzando por la categoría RI, luego se pueden realizar las categorías CA, CRN y VD en paralelo o en el orden que se decida. Cuando finalizan las pruebas se eliminan de las vulnerabilidades encontradas los falsos positivos.

2.4. Etapa 4. Documentación e Informe de los resultados

A la vez que los probadores ejecutan las pruebas van registrando las no conformidades (NC) en el registro de defectos y dificultades (RDD), estas NC no son más que las vulnerabilidades detectadas. El RDD refleja la descripción de las NC y la etapa de

detección. El EP al concluir la jornada de prueba realiza un informe diario del RDD donde recoge todas las NC encontradas. Luego este informe es entregado al equipo de desarrollo y cuando se terminan todas las iteraciones se le entrega el informe final al GS.

4. Conclusiones

Para evaluar el aporte que traería la utilización del procedimiento se encuestaron 7 expertos del GS, de los encuestados 2 evaluaron el procedimiento de regular (Suficientemente bueno con reservas) y los 5 restantes lo evaluaron de bueno (Aplicable con resultados destacados).Este procedimiento además de ser utilizado por probadores de poca experiencia, también puede emplearse en equipos de desarrollo que deseen evaluar las vulnerabilidades de las aplicaciones web que construyen.

El LIPS ha realizado pruebas de seguridad a varias aplicaciones web de la UCI después de aplicado el procedimiento, las NC detectadas fueron entregadas al GS. El proceso de pruebas del GS es muy largo y trabajoso, hasta la fecha ha culminado la revisión de tres aplicaciones. La gráfica 1 muestra la comparación de las NC detectadas por los probadores a tres aplicaciones web antes y después de la utilización del procedimiento, además muestra el tamaño de las aplicaciones en casos de uso. Antes las NC detectadas durante la evaluación de las vulnerabilidades eran inferiores a las detectadas en el GS y después esa cantidad están más igualadas. Analizando estos datos puede concluirse que el procedimiento es útil para los probadores porque después de su uso la cantidad de NC o vulnerabilidades encontradas se asemejan más a las vulnerabilidades explotadas por el GS.

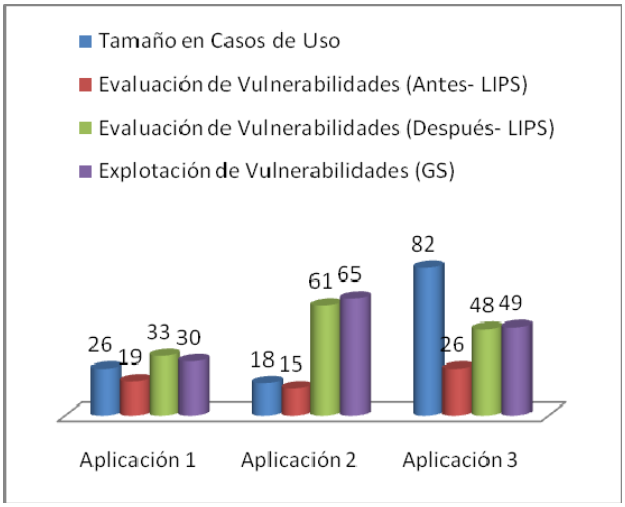

Figura 1. Cantidad de NC antes y después del uso del procedimiento

Referencias

[1] Cenzic, "Web Application Security Trends Report Q3-Q4, 2008", Cenzic, 2009.

[2] OWASP**.** *Owasp testing guide V2.0,* OWASP, 2007.

La madurez de los servicios TI

Antoni Lluís Mesquida, Antònia Mas, Esperança Amengual
Departamento de Matemáticas e Informática, Universitat de les Illes Balears
{antoni.mesquida, antonia.mas, eamengual}@uib.es

Resumen

El interés que la calidad del servicio ha despertado en las organizaciones proveedoras de servicios de Tecnologías de la Información ha propiciado el nacimiento de una nueva disciplina, la gestión de servicios de Tecnologías de la Información. Con el objetivo de centrar la atención, no solamente en el desarrollo de sus productos y/o servicios, sino también en la relación con sus clientes, han ido surgiendo diferentes iniciativas que se analizan este artículo. Algunas de estas iniciativas están relacionadas con la ampliación de los modelos de evaluación y mejora de los procesos de software (CMMI y SPICE), extendiendo estos modelos con nuevos procesos de gestión de servicios. Otras, están basadas en la creación de nuevas normas o estándares específicos de calidad de servicios (ITIL e ISO/IEC 20000).

Palabras clave: Gestión de servicios TI, modelos de madurez, CMMI, ISO/IEC 15504 (SPICE), ISO/IEC 20000.

IT services maturity

Abstract

Interest in quality of service that providers of Information Technology services have shown has favoured the emergence of a new discipline: Information Technology Service Management. With the aim of focusing the attention not only on product development or service provision, but also on the relationship with customers, different initiatives have appeared. In this article these initiatives are analyzed. Some of them are based on the extension of the existent software process assessment and improvement models (such as CMMI and SPICE) by adding new service management processes. Other projects focus on the development of new service quality standards (such as ITIL and ISO/IEC 20000).

Key words: IT service management, maturity models, CMMI, ISO/IEC 15504 (SPICE), ISO/IEC 20000.

Mesquida, A.L., Mas, A. y Amengual, E., "La madurez de los servicios TI", REICIS, vol. 5, no.2, 2009, pp.88-98. Recibido: 22-6-2009; revisado: 6-7-2009; aceptado: 31-7-2009

1. Introducción

Actualmente las organizaciones proveedoras de servicios de Tecnologías de la Información (TI) necesitan disponer de una gestión de servicios efectiva para cumplir las demandas de sus clientes. Para estas organizaciones ya no es suficiente apostar por la mejor tecnología,

una orientación a procesos en el desarrollo de sus productos y en su propia organización interna, sino que también deben considerar la calidad de los servicios que proporcionan a sus clientes. En este sentido, surge una nueva disciplina, la gestión de servicios de Tecnologías de la Información (ITSM, del inglés *Information Technology Service Management*), que se centra en la perspectiva del cliente como principal aporte al negocio.

La gestión de servicios de Tecnologías de la Información es un conjunto de capacidades organizacionales especializadas en proporcionar valor a los clientes en forma de servicios [1]. Para proveer y gestionar de forma eficaz los servicios ofrecidos a lo largo de todo su ciclo de vida, resulta imprescindible definir y adoptar un conjunto de buenas prácticas. Si estas prácticas se agrupan y estructuran en procesos, este conjunto de procesos del área de provisión y gestión de servicios, puede utilizarse para ampliar el concepto de ciclo de vida de procesos de software hacia un ciclo de vida de producto completo que abarque también todos los aspectos relacionados con la provisión y gestión de los servicios.

El estándar internacional ISO/IEC 12207:2008 [2] establece un marco común para los procesos del ciclo de vida del software. Aunque, desde el punto de vista de los autores de este artículo, alguno de los procesos de esta Norma guarda relación con la operación y la provisión del servicio al cliente (ver Tabla 1), para poder abarcar todos los procesos relacionados con los servicios, se han iniciado proyectos de mayor envergadura. Estas iniciativas han surgido en dos direcciones bien distintas. Por una parte, algunas de estas propuestas se centran en el desarrollo de nuevos modelos, normas o estándares específicos de calidad de servicios. Dentro de este grupo, las iniciativas más destacadas son ITIL (*Information Technology Infraestructure Library*) [3-7] e ISO/IEC 20000 [8,9], que se exponen en el segundo apartado de este artículo. Por otra parte, dada la orientación a procesos en la provisión y gestión de los servicios, otros proyectos se centran en la revisión de los estándares de evaluación y mejora de procesos de software, como CMMI (*Capability Maturity Model Integration*) [10] e ISO/IEC 15504 (SPICE) [11,12], ampliándolos para que cubran esta área de servicios. Estas actuaciones se describen en el tercer apartado del artículo.

Proceso	Propósito
Supply Process	Proporcionar un producto o servicio al cliente que cumpla los requisitos acordados.

Proceso	Propósito
Decision Management Process	Seleccionar la mejor opción de entre todas las alternativas posibles en el curso de un proyecto.
Risk Management Process	Identificar, analizar, gestionar y monitorizar los riegos de manera continua.
Configuration Management Process	Establecer y mantener la integridad de todos los resultados de un proyecto o proceso y ponerlos a disposición de todas las partes interesadas.
Information Management Process	Proporcionar información relevante, completa, válida y, si es necesario, confidencial, a todas las partes designadas, durante todo el ciclo de vida del sistema.
Measurement Process	Identificar, analizar y reportar datos de los productos desarrollados y de los procesos implantados en la organización con el objetivo de dar soporte a la gestión efectiva de los procesos y para demostrar de manera objetiva la calidad de los productos.
Software Operation Process	Utilizar el producto software en su entorno operacional y dar soporte a los clientes de este producto software.
Software Maintenance Process	Proporcionar un soporte efectivo en términos de costes para el mantenimiento del producto software entregado.
Software Disposal Process	Retirar un producto software que forma parte de un sistema.
Software Problem Resolution Process	Asegurar que todos los problemas descubiertos son identificados, analizados, gestionados y controlados hasta su resolución.

Tabla 1. Procesos de la norma ISO/IEC 12207:2008 relacionados con la gestión de servicios.

2. Los nuevos estándares para la gestión de servicios

En los subapartados siguientes se introducen los dos estándares de gestión de servicios más conocidos y usados en la actualidad: ITIL e ISO/IEC 20000.

2.1. ITIL (*Information Technology Infrastructure Library*)

ITIL es un conjunto de buenas prácticas de gestión de servicios, desarrollado por la Oficina Gubernamental de Comercio del Reino Unido (OGC, del inglés *Office of Government Commerce*) y aceptado en todo el mundo como estándar de facto. ITIL se centra en la medida continua y en la mejora de la calidad de los servicios ofrecidos, tanto desde la perspectiva del negocio, como desde la perspectiva del cliente.

La versión inicial de ITIL, publicada entre 1989 y 1995, estaba compuesta por 31 libros que cubrían todos los aspectos de la gestión de servicios. Esta versión inicial fue revisada y reemplazada, entre 2000 y 2004 por ITIL V2, formada por sólo siete libros y mejor relacionados. En junio de 2007, ITIL V2 fue sustituida por una versión mejorada y consolidada, ITIL V3. El principal cambio que incorpora ITIL V3 respecto a la versión anterior, es que pasa de una estructura basada en procesos, a una estructura basada en el

ciclo de vida de los servicios. ITIL V3 consta de cinco libros de referencia, cuyos propósitos se muestran en la Tabla 2.

Libro	Propósito
Estrategia del Servicio (*Service Strategy*) [3]	Proporcionar una guía, tanto a los proveedores de servicios de TI como a sus clientes, con la intención de ayudarles a operar y prosperar a largo plazo mediante el establecimiento de una estrategia de negocio bien definida.
Diseño del Servicio (*Service Design*) [4]	Ofrecer pautas para el diseño de servicios apropiados e innovadores, incluyendo su arquitectura, procesos, políticas y documentación, para satisfacer los requisitos de negocio, actuales y futuros, acordados.
Transición del Servicio (*Service Transition*) [5]	Implantar todos los aspectos del servicio, no sólo su aplicación y uso en circunstancias normales. Se debe asegurar que el servicio pueda operar en circunstancias previsibles extremas o anómalas, y que se dispone de un soporte a fallos o errores.
Operación del Servicio (*Service Operation*) [6]	Proveer los niveles de servicio acordados a los usuarios y clientes y gestionar las aplicaciones, tecnología e infraestructura necesaria para dar soporte a la provisión de los servicios.
Mejora continua del servicio (*Continual Service Improvement*) [7]	Evaluar y mejorar de manera continua la calidad de los servicios y la madurez global del ciclo de vida de los servicios y de los procesos subyacentes.

Tabla 2. Libros de referencia de ITIL V3.

Las buenas prácticas de gestión de servicios de ITIL V3 son las que recoge la norma ISO/IEC 20000-2:2005. Si bien esta norma no incluye formalmente el planteamiento de ITIL V3, sí que describe un conjunto integrado de procesos de gestión de servicios que están alineados y son complementarios a los procesos definidos en ITIL V3. Se podría decir que cada uno de los libros de ITIL ofrece una información más ampliada y una guía de buenas prácticas sobre las áreas que se tratan en la norma ISO/IEC 20000.

2.2. La norma ISO/IEC 20000

La norma ISO/IEC 20000 es un estándar de calidad de procesos de gestión de servicios que promueve la adopción de un enfoque de procesos integrados para una provisión eficaz de servicios gestionados que satisfaga los requisitos del negocio y de los clientes. La versión actual de la Norma es del año 2005.

ISO/IEC 20000 está formada por dos partes. La Parte 1 (ISO/IEC 20000-1) [8] define los requisitos para que un proveedor de servicios proporcione servicios gestionados de una calidad aceptable para sus clientes. La Parte 2 (ISO/IEC 20000-2) [9] proporciona una guía para los auditores y ofrece asesoría a los proveedores de servicios para la planificación de las mejoras del servicio.

La Norma se encuentra actualmente bajo un proceso de revisión para alinearse mejor con ITIL V3 y con otros estándares ISO. Se están desarrollando dos nuevas partes, 3 y 4. La Parte 3 (ISO/IEC 20000-3) pretende ofrecer el ámbito para la certificación en entornos en los que el servicio es prestado por múltiples proveedores con diferentes tipos de acuerdos de subcontratación. La parte 4 (ISO/IEC 20000-4) describe un Modelo de Procesos de Referencia (PRM, del inglés, *Process Reference Model*) de gestión de servicios.

La Parte 1 de la Norma define, en los capítulos 3-5, los requisitos de un sistema de gestión y las tareas necesarias para planificar e implementar la gestión de servicios. En esta parte, también se especifican, en los capítulos 6-10, los 13 procesos de provisión de servicios agrupados en cinco categorías. La Figura 1 muestra los procesos y las categorías de proceso.

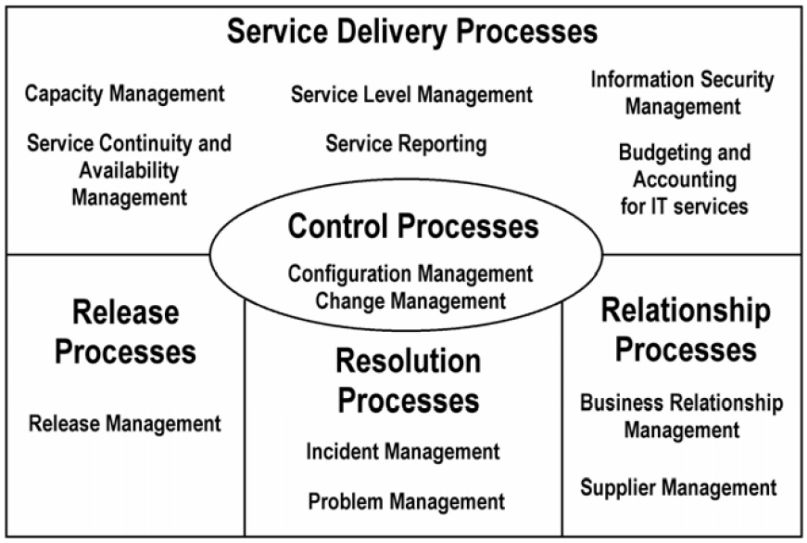

Figura 1. Categorías y procesos de gestión del servicio de la norma ISO/IEC 20000-1.

La Parte 2 de la Norma describe las buenas prácticas para los procesos de gestión de servicios, posibilitando a los proveedores la mejora de la calidad del servicio que proporcionan a sus clientes.

3. La gestión de servicios en los modelos de madurez

Los modelos de madurez de procesos más conocidos y usados, el modelo CMMI del SEI (*Software Engineering Institute*) y el estándar internacional ISO/IEC 15504, han sufrido cambios para contemplar los aspectos relacionados con la gestión de servicios. Así, el

modelo CMMI ha ampliado su modelo de referencia incorporando las áreas específicas de gestión de servicios, mientras que la norma ISO/IEC 15504 se encuentra en proceso de actualización con el objetivo de alinearse con el estándar de gestión de servicios ISO/IEC 20000.

3.1. La ampliación del modelo CMMI

El modelo del SEI específico para la gestión de servicios de TI, *CMMI for Services* (CMMI-SVC) [13] surge en el año 2006. Este modelo, que se describe brevemente en este apartado, es el resultado del proceso de revisión de la arquitectura de CMMI, que se inició en el año 2000 y aún no ha finalizado.

Debido a la utilización del modelo CMMI en diferentes áreas, el modelo fue agrupando sus mejores prácticas dando lugar al concepto de "constelaciones". Una constelación es una colección de componentes CMMI, entre los que se incluyen un modelo, materiales de formación y documentos relacionados con la evaluación, que proporcionan un marco de aplicación específico para un determinado dominio o área de interés.

En un determinado momento de su evolución se empezaron a gestar dos constelaciones nuevas, una constelación específica para la gestión de servicios, *CMMI for Services* (CMMI-SVC), y otra con los procesos específicos de adquisición, *CMMI for Acquisition* (CMMI-ACQ) [14]. Todos los modelos CMMI disponibles hasta la fecha fueron agrupados y considerados como parte de una tercera constelación, *CMMI for Development* (CMMI-DEV) [15], que abarca todos los procesos específicos de desarrollo de software.

Con la publicación de CMMI V1.2, que es la vigente en la actualidad, las tres constelaciones anteriormente citadas cobraron autonomía propia, dando lugar a tres modelos diferentes de CMMI. La Figura 2 muestra las áreas de proceso (PA, del inglés *Process Area*) de las tres constelaciones de CMMI V1.2.

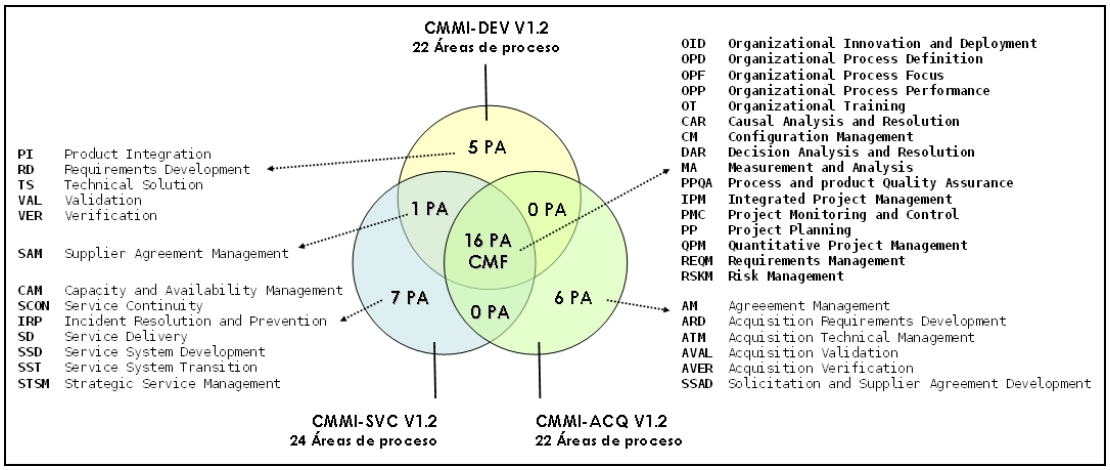

Figura 2. Constelaciones de CMMI V1.2.

La constelación orientada a servicios, CMMI-SVC V1.2, contiene 24 áreas de proceso (ver Tabla 3). 16 son compartidas por las tres constelaciones y forman lo que se conoce como el *CMMI Foundation Model* (CMF). 7 son específicas de gestión de servicios. El área de proceso restante, *Supplier Agreement Management* (SAM), es compartida con CMMI-DEV.

Categoría	Área de proceso (PA)
Process Management	OID *Organizational Innovation and Deployment* OPD *Organizational Process Definition* OPF *Organizational Process Focus* OPP *Organizational Process Performance* OT *Organizational Training*
Support	CAR *Causal Analysis and Resolution* CM *Configuration Management* DAR *Decision Analysis and Resolution* MA *Measurement and Analysis* PPQA *Process and product Quality Assurance*
Project Management	CAM *Capacity and Availability Management* IPM *Integrated Project Management* PMC *Project Monitoring and Control* PP *Project Planning* QPM *Quantitative Project Management* REQM *Requirements Management* RSKM *Risk Management* SAM *Supplier Agreement Management* SCON *Service Continuity*
Service Establishment and Delivery	IRP *Incident Resolution and Prevention* SD *Service Delivery* SSD *Service System Development* SST *Service System Transition* STSM *Strategic Service Management*

Tabla 3. Áreas de Proceso (PA) del modelo CMMI-SVC V1.2.

3.2. La ampliación del modelo ISO/IEC 15504

La norma ISO/IEC 15504 está siendo ampliada para cubrir los procesos de gestión de servicios. En ese sentido, y aunque no se dispone aún de información precisa de su contenido, el subcomité JTC 1/SC 7, responsable del estándar ISO/IEC 15504, trabaja en la nueva Parte 8: ISO/IEC NP TR 15504-8: *An exemplar process assessment model for IT service management* [16]. Esta parte incluirá un Modelo de Evaluación de Procesos de gestión de servicios, basado en el Modelo de Procesos de Referencia que se definirá en la nueva parte ISO/IEC CD TR 20000-4 [17].

Aún así y según la información aparecida en el SPICE User Group [18], se ha producido la aprobación por parte de la ISO de una nueva generación de estándares, denominada serie 31000 (SPICE 31K), que sustituirá a las diferentes partes del estándar ISO/IEC 15504 y que incluirá algunas partes nuevas. Concretamente, se prevé que la Parte 15504-8 se convierta en la futura Parte 31062 que se denominará *Process Assessment Model for IT Service Management Processes*.

4. Conclusiones

Desde la aparición de ITIL e ISO/IEC 20000 ya se dispone de estándares de calidad específicos para la gestión de servicios, que proporcionan un modelo de procesos de referencia de gestión de servicios. Sin embargo, para las organizaciones proveedoras de servicios de Tecnologías de la Información resulta, además, muy importante disponer de modelos de evaluación y mejora que les permitan conocer el nivel de madurez de sus procesos de gestión de servicios.

La necesidad de evaluar todos los procesos que se realizan en una organización proveedora de servicios de TI propició la ampliación de los modelos de madurez de procesos (CMMI e ISO/IEC 15504) con los procesos de gestión de servicios. A consecuencia de ello, el SEI ha ampliado el modelo CMMI incorporando una constelación con los procesos específicos para la gestión de servicios, CMMI-SVC, liberada en febrero de 2009. Desde la ISO se está trabajando en la misma línea, desarrollando tanto un modelo de referencia (futura parte ISO/IEC 20000-4) como uno de evaluación de procesos de gestión de servicios (futura parte ISO/IEC 15504-8).

En este artículo se ofrece una visión de la situación actual de los estándares de gestión de servicios con la intención, tanto de presentar los modelos existentes, como de esclarecer la posible confusión que pueda causar la presencia de diversos modelos con un objetivo común, ante un público no familiarizado con las disciplinas de gestión de calidad. En la Tabla 4 se muestra una clasificación de los diferentes estándares enumerando algunas de sus características.

Clasificación / Característica	Específicos de Gestión de servicios		Ampliación de modelos de madurez	
	ITIL	ISO/IEC 20000	CMMI SVC	ISO/IEC 15504-8
Desarrollador	*Office of Government Commerce* (OGC)	*International Organization for Standardization* (ISO)	*Software Engineering Institute* (SEI)	*International Organization for Standardization* (ISO)
Sitio web oficial	http://www.itil-officialsite.com	http://www.iso.org	http://www.sei.cmu.edu/cmmi	http://www.iso.org
Carácter	Privado	Público	Privado	Público
Versión vigente	V 3	2005	V 1.2	No publicado
Fecha de aparición de la versión vigente	Junio 2007	Diciembre 2005	Febrero 2009	No publicado
Arquitectura del estándar	5 libros. Cada uno representa un área del ciclo de vida del servicio.	2 partes ya publicadas y 2 nuevas partes en proceso de elaboración.	CMMI-SVC es una de las tres constelaciones del modelo CMMI.	ISO/IEC 15504-8 se encuentra, junto con la parte 9, en proceso de elaboración.
Procesos que abarca cada norma	27 procesos	13 procesos, agrupados en 5 categorías	24 áreas de proceso, agrupadas en 4 categorías	Pendientes de definición
Modelo de certificación acreditado	Certificación de profesionales: • *Foundation level* • *Intermediate level* • *ITIL Expert* • *ITIL Master*	Certificación de empresas. Evaluación por evaluadores acreditados. Pendiente de definición en España.	Certificación de empresas. Evaluación por evaluadores acreditados por el SEI.	Certificación de empresas. Evaluación por evaluadores acreditados. Pendiente de definición.

Tabla 4. Principales características de los diferentes estándares de gestión de servicios.

El creciente interés de las organizaciones en evaluar sus procesos, tanto los de desarrollo como los de gestión de servicios, ha impulsado diferentes iniciativas para el desarrollo de modelos de aplicación simultánea de estándares de calidad. En esta línea, el centro de investigación luxemburgués Henri Tudor ha creado el modelo AIDA [19] para la implantación conjunta de las normas ISO/IEC 15504 e ITIL. Por otra parte, nuestro grupo de investigación Miprosoft ya desarrolló en el proyecto QuaSAR I [20] un modelo de

implantación simultánea de las normas ISO/IEC 15504 e ISO 9001. Siguiendo en el campo de la mejora de procesos y dado que la tendencia actual en las organizaciones se dirige hacia la gestión de servicios, Miprosoft está trabajando actualmente en la creación de un nuevo modelo que combine la implantación de las normas ISO/IEC 15504 e ISO/IEC 20000.

Agradecimientos

Esta investigación ha sido posible gracias al soporte ofrecido por el proyecto coordinado SOAQTest: Calidad en los procesos de desarrollo y pruebas en arquitecturas orientadas a servicios (TIN2007-67843-C06-04).

Referencias

[1] Laboratorio Nacional de Calidad del Software, *Guía Avanzada de Gestión de Servicios*. Instituto Nacional de Tecnologías de la Comunicación (INTECO), 2008.

[2] International Organisation for Standardization, *ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes*, ISO/IEC, 2008.

[3] Office of Government Commerce, *ITIL Version 3 Service Strategy,* OGC, 2007.

[4] Office of Government Commerce, *ITIL Version 3 Service Design,* OGC, 2007.

[5] Office of Government Commerce, *ITIL Version 3 Service Transition,* OGC, 2007.

[6] Office of Government Commerce, *ITIL Version 3 Service Operation,* OGC, 2007.

[7] Office of Government Commerce, *ITIL Version 3 Continual Service Improvement,* OGC, 2007.

[8] International Organisation for Standardization, *ISO/IEC 20000-1:2005 Information Technology – Service Management – Part 1: Specification*, ISO/IEC, 2005.

[9] International Organisation for Standardization, *ISO/IEC 20000-2:2005 Information Technology – Service Management – Part 2: Code of practice*, ISO/IEC, 2005.

[10] Software Engineering Institute, *Capability Maturity Model® Integration (CMMI®) Version 1.2 Overview* (http://www.sei.cmu.edu/cmmi/adoption/pdf/cmmi-overview07.pdf, Junio 2009), Carnegie Mellon University, 2007.

[11] International Organisation for Standardization, *ISO/IEC 15504-2:2004 Software Engineering – Process Assessment – Part 2: Performing an assessment*, ISO/IEC, 2003.

[12] International Organisation for Standardization, *ISO/IEC 15504-1:2004 Information Technology – Process Assessment – Part 1: Concepts and Vocabulary*, ISO/IEC, 2004.

[13] Software Engineering Institute, *CMMI-SVC, CMMI® for Services (CMMI-SVC) Version 1.2*, SEI, 2009.

[14] Software Engineering Institute, *CMMI-ACQ, CMMI® for Acquisition (CMMI-ACQ) Version 1.2*, SEI, 2007.

[15] Software Engineering Institute, *CMMI-DEV, CMMI® for Development (CMMI-DEV) Version 1.2*, SEI, 2006.

[16] International Organisation for Standardization, *ISO/IEC NP TR 15504-8 Information technology – Software process assessment – Part 8: An exemplar process assessment model for IT service management*, ISO/IEC, 2009.

[17] International Organisation for Standardization, *ISO/IEC CD TR 20000-4 Information technology – Service Management – Process Reference Model*, ISO/IEC, 2009.

[18] *The SPICE User Group* (http://www.spiceusergroup.org/forum2/topics/next-generation-15504-the, Junio 2009).

[19] Barafort B., Di Renzo B. y Merlan O., "Benefits resulting from the combined use of ISO/IEC 15504 with the Information Technology Infrastructure Library (ITIL)", *Proceedings of the International Conference PROFES'2002. Finland*, 2002.

[20] Mas, A. y Amengual, E., "Un nuevo método para la aplicación simultánea de ISO/IEC 15504 y ISO 9001:2000 en PYMES de desarrollo de software", *Novática*, vol. 164, Agosto 2004, pp. 24-31, 2004.

Una aplicación de ISO/IEC 15504 para la evaluación por niveles de madurez de PYMEs y pequeños equipos de desarrollo

Javier Garzás
Kybele Consulting
javier.garzas@kybeleconsulting.com
y Universidad Rey Juan Carlos
javier.garzas@urjc.es
Carlos Manuel Fernández
AENOR
cmfernandez@aenor.es
Mario Piattini
Universidad de Castilla – La Mancha
mario.piattini@uclm.es

Resumen

La calidad del software está tomando mayor importancia en las organizaciones por su influencia en los costes finales y como elemento diferenciador de la competencia y de la imagen frente a sus clientes. En este sentido muchas organizaciones están implantando modelos de mejora de procesos software. Sin embargo, la implantación en PYMEs de los modelos referentes en la actualidad, CMMI e ISO 15504, supone una gran inversión en dinero, tiempo y recursos. En este artículo se presenta una adaptación de ISO 15504 para la evaluación por niveles de madurez en PYMEs y pequeños equipos de desarrollo.

Palabras clave: ISO/IEC 15504, mejora de procesos software, calidad del software, PYMEs, CMMI, pequeños equipos.

Application of ISO/IEC 15504 for maturity assessment of SME and small development teams

Abstract

Software quality is becoming more important in organizations because of its influence on the final costs, image and differentiation. In this way many organizations are implementing software process improvement models. However, the implementation in SMEs of the referent models, CMMI and ISO 15504, is a great investment in cost, time and resources. This paper presents an adaptation of ISO 15504 for the assessment of maturity levels in SMEs and small development teams.

Palabras clave: ISO/IEC 15504, software process improvement, software quality, SMEs, CMMI, small teams.

Garzas, J., Fernández, C.M. y Piattini, M., "Una aplicación de ISO/IEC 15504 para la evaluación por niveles de madurez de PYMEs y pequeños equipos de desarrollo", REICIS, vol. 5, no.2, 2009, pp.88-98. Recibido: 22-6-2009; revisado: 6-7-2009; aceptado: 31-7-2009

1. Introducción

La calidad del software está tomando cada vez mayor importancia en las organizaciones por su influencia en los costes finales, como elemento diferenciador de la competencia y de imagen para clientes, más aún con el crecimiento de las fábricas software [1]. Si bien, en la actualidad, diversos estudios como el realizado por la Asociación de Técnicos en Informática (ATI) en el ámbito español [2], continúan mostrando que el mercado está poco maduro en el control de calidad software. Por este motivo, muchas organizaciones están implantando modelos de mejora de procesos. Y más concretamente, como muestra el estudio elaborado por INTECO [3], de entre todos los modelos de mejora de procesos dos se han convertido en los de mayor uso en la industria del software: CMMI –DEV [4] e ISO 15504 SPICE [5].

En la actualidad, tanto CMMI-DEV como ISO/IEC 15504 con la publicación de la parte 7 "Assessment of Organizacional Maturity", incorporan el tipo de evaluación más extendida en la industria del software, por niveles de madurez, permitiendo dar una puntuación cuyo alcance es la organización (departamento, proyecto, etc.).

Sin embargo numerosos estudios [6-8], confirman que CMMI e ISO/IEC 15504 están orientados a grandes organizaciones y no abordan explícitamente las necesidades de las PYMEs y de los pequeños grupos y equipos de desarrollo, donde la aplicación de estos modelos resulta costosa en términos económicos y de esfuerzo, ya que requieren una gran inversión en dinero, tiempo y recursos, sus recomendaciones son complejas de aplicar y el retorno de la inversión se produce a muy largo plazo. Y en este sentido se han identificado varias iniciativas nacionales e internacionales orientadas expresamente a la PYME; entre las iniciativas más conocidas se pueden destacar el ESSI (European Software and System Initiative) en la Unión Europea, los modelos MoProSoft [9] y EvalProSoft [10] en México, el modelo ITMARK del ESI (Instituto Europeo del Software) y el proyecto COMPETISOFT para Iberoamérica [11, 12].

En este artículo se presenta el modelo desarrollado por el grupo formado por AENOR, Universidad de Castilla – La Mancha, Universidad Rey Juan Carlos, Kybele Consulting y Prysma en la elaboración de un modelo de evaluación de procesos en PYMEs y pequeños grupos de desarrollo por niveles de madurez y según la norma ISO/IEC 15504.

El modelo cumple con la ISO/IEC 17021[3], y por su naturaleza dicho modelo estaría fácilmente alineado con las guías ISO/IEC 29110 (Lifecycle Profiles for Very Small Enterprises) [13] que se están elaborando actualmente, y con otras normas muy importantes en el sector como son la ISO/IEC 27001 y la ISO/IEC 20000, mencionadas en importantes iniciativas como la hoja de ruta de AENOR [14].

En la sección 2 se describen los niveles de madurez y el modelo de procesos establecido, en la sección 3 se presenta el modelo de evaluación de procesos, en la sección 4 se especifican los requisitos establecidos para llevar a cabo la auditoría, y en la sección 5 se muestran las conclusiones.

2. Los niveles de madurez y el modelo de procesos

La norma ISO/IEC 15504-7 describe las bases para llevar a cabo evaluaciones por niveles de madurez, para lo cual muestra un conjunto de niveles y procesos asociados, tomando como base la norma ISO 12207:1995 / Amd 1:2002 y Amd. 2: 2004, si bien actualmente está disponible la versión ISO 12207:2008.

Nivel de Madurez 2	Nivel de Madurez 3
Proceso de suministro	Proceso de gestión de la decisión
Proceso de gestión del modelo del ciclo de vida	Proceso de gestión de riesgos
Proceso de planificación del proyecto	Proceso de gestión de infraestructuras
Proceso de evaluación y control del proyecto	Proceso de gestión de recursos humanos
Proceso de gestión de la configuración	Proceso de análisis de requisitos del software
Proceso de medición	Proceso de diseño de la arquitectura del software
Proceso de definición de requisitos de stakeholders	Proceso de integración del software
Proceso de análisis de los requisitos del sistema	Proceso de verificación del software
Proceso de gestión de la configuración del software	Proceso de validación del software
Proceso de aseguramiento de la calidad del software	Proceso de diseño de la arquitectura del sistema
	Proceso de integración del sistema

Tabla 1. Niveles de madurez y modelo de procesos

En el modelo que aquí se presenta, los procesos definidos para los diferentes niveles de madurez están en línea a como lo hace ISO/IEC 15504-7 pero tomando como base la nueva ISO 12207:2008, versión más reciente, adaptándolo a pequeños grupos de desarrollo

[3] ISO/IEC 17021:2006 (Conformity assessment -- Requirements for bodies providing audit and certification of management systems)

y PYMEs, de ahí que los procesos definidos en este modelo para los niveles de madurez difieren de los definidos en la norma ISO/IEC 15504-7. En concreto, esta primera versión del modelo establece 6 niveles de madurez para clasificar a las organizaciones, desde el 0 (nivel inferior) hasta el 5 (superior). En la Tabla 1 se muestra el conjunto de procesos definidos para los niveles de madurez 2 y 3.

3. El modelo de evaluación

En la Figura 6 se muestra un resumen de los diferentes componentes del modelo de evaluación, la relación entre ellos y la obligatoriedad de su implementación.

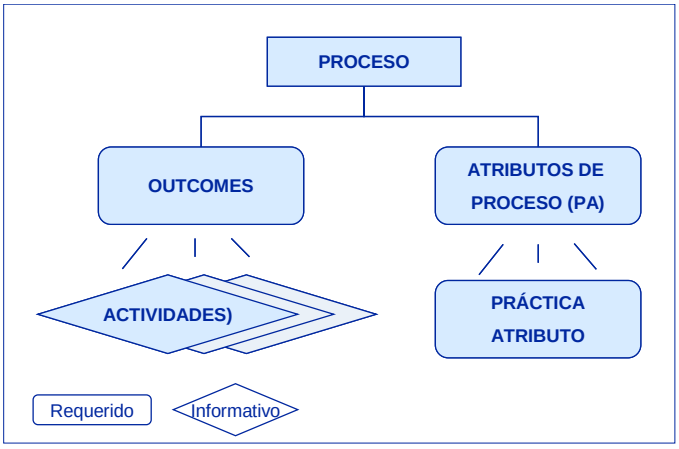

Figura 6. Componentes del modelo de evaluación

NIVEL DE MADUREZ	DESCRIPCIÓN
Nivel de madurez 0	La organización no tiene una implementación efectiva de los procesos.
Nivel de madurez 1	Los procesos objeto de evaluación alcanzan el nivel de capacidad 1, es decir, existen productos resultantes para los mismos y el proceso se puede identificar.
Nivel de madurez 2	Los procesos del nivel de madurez 2 tienen nivel de capacidad 2 o superior.
Nivel de madurez 3	Los procesos de los niveles de madurez 2 y 3 tienen nivel de capacidad 3 o superior.
Nivel de madurez 4	Uno o más procesos tienen nivel de capacidad 4 o superior.
Nivel de madurez 5	Uno o más procesos tienen nivel de capacidad 5.

Tabla 2. Reglas de derivación para los niveles de madurez

Para que una organización pueda alcanzar un nivel de madurez, se debe determinar el nivel de capacidad[4] de los procesos correspondientes al nivel de madurez, y con el nivel de capacidad, se derivará un nivel de madurez, de acuerdo a unas reglas[5] (ver *"Reglas de derivación"* en la Tabla 2).

Para medir la capacidad de un proceso, se utiliza un conjunto de atributos de proceso (PAs)[6], donde cada atributo define un aspecto particular de capacidad de proceso, tal y como se muestra en la Tabla 3[7]. En este sentido, los atributos de proceso son comunes para todos los procesos y describen las características que deben estar presentes para institucionalizar un proceso.

Nivel de capacidad	Atributo de proceso (PA)
Nivel 1: Proceso Realizado	PA 1.1 Realización del proceso
Nivel 2: Proceso Gestionado	PA 2.1 Gestión de la realización PA 2.2 Gestión del producto de trabajo
Nivel 3: Proceso Establecido	PA 3.1 Definición del proceso PA 3.2 Despliegue del proceso
Nivel 4: Proceso Predecible	PA 4.1 Medición del proceso PA 4.2 Control del proceso
Nivel 5: Proceso en optimización	PA 5.1 Innovación del proceso PA 5.2 Optimización continua

Tabla 3. Niveles de capacidad y atributos de proceso

Asimismo, el cumplimiento de los atributos de proceso determinará el nivel de capacidad del proceso, y de ahí el nivel de madurez vendrá determinado por los niveles de capacidad de todos los procesos asociados al nivel de madurez. La Tabla 4 muestra las equivalencias entre los niveles de capacidad y los niveles de madurez correspondientes a la propuesta de adaptación.

En concreto, para determinar el cumplimiento de un atributo de proceso, cada uno de ellos tiene definido un conjunto de prácticas que indican qué se debe realizar para alcanzar el propósito de dicho atributo de proceso. Estas prácticas son conocidas como "prácticas

[4] La capacidad es una evaluación de la calidad un proceso, de manera aislada.
[5] Coforme a la norma ISO/IEC 15504-7:2008.
[6] PAs: de la terminología inglesa "Process Attributes", especificado en la norma ISO/IEC 15504.
[7] Conforme a la norma ISO/IEC 15504-2:2003.

atributo". A modo de ejemplo, las prácticas atributo asociadas al atributo de proceso "PA 2.1 Gestión de la Realización" son: 1) definir los objetivos del proceso, 2) planificar y controlar el proceso, 3) Adaptar la realización del proceso, 4) asignar las responsabilidades y autoridades, 5) Asignar los recursos y la información para el proyecto y 6) Gestionar la comunicación entre las partes involucradas.

		NIVELES DE CAPACIDAD				
		1	2		3	
		PA 1.1	PA 2.1	PA 2.2	PA 3.1	PA 3.2
PROCESOS DEL NIVEL DE MADUREZ 2	Proceso de suministro	Objetivo para la consecución del nivel de madurez 2				
	Proceso de gestión del modelo de ciclo de vida					
	Proceso de planificación del proyecto					
	Proceso de evaluación y control del proyecto					
	Proceso de gestión de la configuración					
	Proceso de medición					
	Proceso de definición requisitos de stakeholders					
	Proceso de análisis de los requisitos del sistema					
	Proceso de gestión de la configuración software					
	Proceso de aseguramiento de la calidad software					
PROCESOS DEL NIVEL DE MADUREZ 3	Proceso de gestión de la decisión		Objetivo para la consecución del nivel			
	Proceso de gestión de riesgos					
	Proceso de gestión de infraestructuras					
	Proceso de gestión de recursos humanos					
	Proceso de análisis de requisitos del software					
	Proceso de diseño de la arquitectura del software					
	Proceso de integración del software					
	Proceso de verificación del software					
	Proceso de validación del software					
	Proceso de diseño de la arquitectura del sistema					
	Proceso de integración del sistema					

Tabla 4. Equivalencias entre los niveles de capacidad y los niveles de madurez 2 y 3

Por otro lado, los procesos además de disponer de partes comunes (los atributos de proceso y las prácticas atributo), disponen de partes específicas, conocidas como *outcomes*[8] y actividades. Un *outcome*, por tanto, aplica a un proceso y describe las características únicas que deben implementarse para satisfacer dicho proceso. Al igual que los atributos de proceso y las prácticas atributo, son componentes requeridos.

Los *outcomes* son requeridos para el atributo de proceso "PA 1.1 Realización del proceso", siendo por tanto la implementación de los *outcomes* del proceso la evidencia del alcance o logro de dicho atributo.

Para la interpretación e implementación de los *outcomes*, cada proceso proporciona un conjunto de descripciones detalladas. Estas descripciones son conocidas como actividades. Como se observa en la Figura 6 pág. 91, las actividades son un componente informativo.

Por último destacar que tanto los *outcomes* como las actividades de cada proceso se encuentran definidos en el modelo de procesos de referencia, es decir, en la norma ISO/IEC 12207:2008, mientras que los atributos de proceso se especifican en la parte 2 de la norma ISO/IEC 15504 y las prácticas atributo se corresponden con una adaptación de la parte 5 de la norma ISO/IEC 15504.

4. Requisitos de la auditoría

Conforme al modelo presentado, se han definido una serie de requisitos para llevar a cabo la auditoría. Estos requisitos se han dividido en dos grupos: elementos y criterios de calificación. Además, el modelo cumple con la norma ISO/IEC 17021.

4.1. Elementos de la auditoría

El elemento principal de las auditorías son las evidencias de implementación de los procesos. En este sentido, se debe destacar que para alcanzar un nivel de madurez, la organización debe presentar evidencia objetiva de cada uno de los atributos de proceso de todos los procesos dentro del nivel de madurez[9], en concreto se presentará "evidencia

[8] Se ha mantenido la terminología inglesa de forma que esté alineado con la norma ISO/IEC 12207:2008.

[9] Conforme a la norma ISO/IEC 15504-2:2003 e ISO/IEC 15504-7:2008.

objetiva" de cada uno de los *outcome*s y de las prácticas atributo de todos los procesos asociados al nivel de madurez[10].

Una evidencia objetiva debe estar formada por un documento donde se evidencie que en los procesos de la organización la práctica atributo o el *outcome* se encuentra documentado, y por una serie de indicadores que evidencien la implementación de dicho componente. Estos indicadores son conocidos como indicadores de implementación.

Los indicadores de implementación pueden ser de 3 tipos[11]: 1) Artefactos directos, salidas que resultan de la implementación directa de un *outcome* o de una práctica atributo; 2) Artefactos indirectos, artefactos que son consecuencia de la implementación de un *outcome* o de una práctica atributo, pero que no son el propósito para el cual se realizan; 3) Afirmaciones, entrevistas que confirman la implementación de un *outcome* o de una práctica atributo.

En este sentido, una evidencia objetiva se compone de un documento que describa el proceso, más un artefacto directo (una evidencia del resultado de la aplicación del proceso en proyectos), más un artefacto indirecto (por ejemplo el acta de una reunión en que se tratase el proceso, un plan de proyecto en el que se planificase el proceso, etc.) y/o una afirmación (corroboración oral por parte de miembros del equipo).

EVIDENCIA OBJETIVA = DOCUMENTO AND (ARTEFACTO DIRECTO AND (ARTEFACTO INDIRECTO OR AFIRMACION))

Figura 7. Evidencia objetiva

Además del elemento relativo a las evidencias de implementación de los procesos, para realizar la auditoría previamente se debe identificar la "*muestra de proyectos"* sobre los que se realizará la evaluación y se debe formar un equipo auditor.

En el contexto de la muestra de proyectos, la organización deberá seleccionar al menos 4 proyectos que evidencien los procesos del nivel de madurez objeto de la evaluación[12].Y respecto al equipo auditor, el principal requisito es que deberá estar compuesto como mínimo por 4 auditores: 1 auditor jefe, 1 auditor y 2 auditores internos, siendo los auditores internos miembros de la organización sobre la cual se va a realizar la auditoría.

[10] Adaptación particular.
[11] De manera similar a como se realiza en otros modelos de mejora de procesos, como por ejemplo CMMI.
[12] Conforme a la norma ISO/IEC 15504-7:2008.

4.2. Criterios de calificación

El segundo grupo de requisitos para la auditoría son los criterios de evaluación de cada uno de los componentes del modelo de evaluación. La calificación de los atributos de proceso dependerá de la calificación que hayan obtenido sus prácticas atributo asociadas y sus *outcome*s en el caso del atributo de proceso PA 1.1.[13]

- *Not Achieved* (N): El grado de alcance de los componentes asociados al atributo de proceso es del 0% al 15%.
- *Partially Achieved* (P): El grado de alcance de los componentes asociados al atributo de proceso es del 16% al 50%.
- *Largely Achieved* (L): El grado de alcance de los componentes asociados al atributo de proceso es del 51% al 85%.
- *Full Achieved* (F): El grado de alcance de los componentes asociados al atributo de proceso es del 86% al 100%.

Una vez calificados los atributos de proceso, se califica el nivel de capacidad de cada proceso. En concreto, para alcanzar un nivel de capacidad, los atributos de proceso de los niveles inferiores deben estar calificados como *Fully Achieved,* y los atributos de proceso del nivel de capacidad que está siendo evaluado deben estar calificados como *Largely Achieved* o *Fully Achieved*. Por ejemplo, para que el proceso evaluado alcance el nivel de capacidad 1, su atributo de proceso PA 1.1 debe tener la calificación de *Largely Achieved* o *Fully Achieved* y para alcanzar el nivel de capacidad 2 el atributo de proceso PA 1.1 debe tener la calificación como *Fully Achieved* y sus atributos de proceso PA 2.1 y PA 2.2 deben tener la calificación *Largely Achieved* o *Fully Achieved*.

Por último, el nivel de madurez se calificará en base a los niveles de capacidad obtenidos para el conjunto de procesos correspondientes a dicho nivel. Las reglas de derivación se detallan en la Tabla 2, pág.91.

5. Conclusiones

En este artículo se ha presentado un modelo para la evaluación de procesos por niveles de madurez en PYMEs y pequeños grupos según la norma ISO/IEC 15504. El modelo ha sido desarrollado por un grupo de trabajo formado por AENOR, Universidad de Castilla – La

[13] Conforme con la norma ISO/IEC 15504-2.

Mancha, Universidad Rey Juan Carlos, Kybele Consulting y Prysma, que han aportado su experiencia profesional a la elaboración del modelo. El modelo será aplicado durante este año 2009 en la evaluación de un grupo de 16 Pymes. También se está desarrollando el portal www.iso15504.es para centralizar información del modelo.

El principal objetivo de la propuesta es minimizar los problemas que en la actualidad PYMEs y pequeños grupos tienen con modelos de mejora de procesos más orientados a grandes organizaciones y además que la evaluación esté claramente orientada y adaptada a procesos software, por lo que la complejidad y coste de la implantación del modelo y la de su evaluación es menor. A diferencia de la norma ISO/IEC 15504-7, la propuesta utiliza la última versión del modelo de procesos ISO/IEC 12207. El modelo cumple con la ISO/IEC 17021, y estaría fácilmente alineado con las guías ISO/IEC 29110 [13] y con otras como ISO/IEC 27001 o 20000.

6. Referencias

[1] Piattini M., Garzás J., *Fábricas de software: Experiencias, tecnologías y organización*, Ra-Ma, 2007.

[2] ATI, "Estudios sobre la práctica de las pruebas de software en España". (http://www.idg.es/cio/Escaso_interes_de_las_empresas_TI_espa%C3%B1olas_por_probar_la_calidad_del_software/doc71182-management.htm. visitado en Febrero 2009). 2008.

[3] INTECO. *Estudio sobre la certificación de la calidad como medio para impulsar la industria de desarrollo del software en España* (http://www.inteco.es/Calidad_del_Software/estudios_e_indicadores/publicaciones/calidad_sw_estudios_e_informes/Calidad_software_32, consultado por última vez en Abril de 2009). 2008.

[4] Chrissis M.B., Konrad M., Shrum S., *CMMI: Guidelines for Process Integration and Product Improvement*, Addison-Wesley Professional, 2006.

[5] ISO. *ISO/IEC 15504-2:2003, Information technology— Process assessment — Part 2: Performing an assessment*. International Standards Organization, 2004.

[6] Saiedian H, Carr N. Characterizing a software process maturity model for small organizations. , 1997. Vol. pp. . ACM SIGICE Bulletin, vol. 23, 1997. p.2.

[7] Staples M, Niazi M, Jeffery R, Abrahams A, Byatt P, Murphy R., "An exploratory study of why organizations do not adopt CMMI", *Journal of Systems and Software*, vol.80, nº 6, pp. 883-895, 2007.

[8] Hareton L., Terence Y., "A Process Framework for Small Projects", *Software Process Improvement and Practice*, Vol. 6, nº 2, pp. 67-83, 2001.

[9] Oktaba H., Esquivel C., Su Ramos A., Martínez A., Quintanilla G., Ruvalcaba M., López F., Rivera M., Orozco M., Fernández Y., Flores M., *Software Industry Process Model MoProSoft Version 1.3.2* (http://www.comunidadmoprosoft.org.mx/COMUNIDAD_MOPROSOFTADM/Documentos/V_1.3.2_MoProSoft_English.pdf, visitado en Junio de 2009). vol. 2009. México, 2006.

[10] Oktaba H., Esquivel C., Su Ramos A., Palacios J., Pérez C.J., López F., *Método de Evaluación de procesos para la industria de software EvalProSoft Versión 1.1* (http://www.software.net.mx/NR/rdonlyres/ED7B3399-0CA4-412E-9FAC-0EEB94F85C5F/1224/EvalProSoftv11.pdfEvalprosoft, visitado en Junio de 2009). vol. 2009. México, 2004.
[11] Piattini M., Oktaba H., Orozco M.J., Alquicira C., *COMPETISOFT. Mejora de Procesos Software para Pequeñas y Medianas Empresas y Proyectos*. Ra-ma, 2008.
[12] Oktaba H., García F., Piattini M., Ruiz F., Pino F.J., Alquicira C., "Software Process Improvement: The Competisoft Project", *IEEE Computer*, vol. 40, nº 10, pp.21-28, 2007.
[13] Calvo-Manzano J.A., Garzás J., Piattini M., Pino F.J., Salillas J., Sánchez J.L., "Perfiles del ciclo de vida del software para pequeñas empresas: Los informes técnicos de ISO/IEC 29110". X Jornadas de innovación y calidad del software (JICS). Conferencia Iberoamericana de calidad del software. Madrid, 2008.
[14] Fernández C.M. "Aenor establece su hoja de ruta de certificaciones TIC". Computerworld, vol. 28, 2009.