Revista
Española de

Innovación, Calidad e Ingeniería del Software

REICIS

Volumen 4, Número 2 (especial X JICS), septiembre, 2008

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

Publicado por la Asociación de Técnicos de Informática
www.ati.es

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

Contenidos

REICIS

Editorial

REICIS

El grupo de Calidad del Software de ATI ha consolidado su posición como principal promotor de la disciplina de ingeniería y calidad del software con la décima edición de las Jornadas sobre Innovación y Calidad del Software (las tradicionales JICS). Estas X JICS pretenden además potenciar 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 por lo que constituyen la promoción de una I Conferencia Iberoamericana de Calidad del Software (CICS). Por otra parte, las X JICS incorporan la presencia de la ponencia de un destacado experto europeo en la disciplina de ingeniería de software como es Darren Dalcher, Director del UK National Centre for Project Management en la Middlesex University y editor de la revista Software Process Improvement and Practice.

Por otra parte, queremos resaltar la línea de calidad de los trabajos, eminentemente prácticos pero rigurosos, aceptados entre los remitidos en la convocatoria de contribuciones: las ponencias aceptadas (con una tasa de rechazo del 40%) han sido sometidos a un completo proceso de revisión por el comité de programa así como a una cuidadosa labor de revisión de estilo, de terminología y de ortotipografía para garantizar el mejor resultado para nuestros lectores. Por supuesto, no cabe olvidar el apoyo de los patrocinadores (Telelogic, Steria, Deiser, GESEIN y SOGETI) no sólo aportando recursos sino también interesantes presentaciones de experiencias prácticas de sus expertos. Los debates promovidos en las mesas redondas así como la promoción de las actividades de comunicación y *networking* entre los participantes, tanto a nivel presencial como a través de la lista de distribución, los medios electrónicos y la nueva oferta formativa con plataforma *e-learning*. 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 la trayectoria pionera en España que, desde 1997, está proporcionando, a través de la Asociación de Técnicos de Informática, el apoyo para la productividad y la calidad en los proyectos de software. Este perfil ha sido reconocido por el apoyo del Ministerio de Industria, Turismos y Comercio con su apoyo institucional dentro de la convocatoria de la orden ITC/390/2007. Por último, debemos resaltar la aportación de datos de gran importancia no sólo mediante los eventos organizados sino también a través de la realización de estudios específicos (por ejemplo, sobre las prácticas de pruebas, el diseño de casos y los factores que dificultan su implantación eficiente y eficaz en las organizaciones) que permiten un mejor conocimiento de la práctica real de esta disciplina en España.

Luis Fernández Sanz
Juan J. Cuadrado-Gallego
Editores

Presentación

REICIS

En este número especial de septiembre de 2008 de REICIS, por primera 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 décima edición desde su inicio en 1998. En esta ocasión, el Grupo de Calidad del Software de ATI (www.ati.es/gtcalidadsoft) no sólo ha querido cumplir con esta decena de ediciones sino que ha apostado por una apertura a nuevos retos como la presencia de eminentes ponentes invitados de gran presencia internacional y la potenciación de los vínculos iberoamericanos para convertir a este evento en la referencia sobre calidad del software en la amplia comunidad latina. Los trabajos aceptados han sido sometidos a un completo proceso de revisión por el comité de programa así como a una cuidadosa labor de revisión de estilo, terminología y ortotipografía para garantizar la mejor calidad para nuestros lectores. Este número especial constituye en definitiva la publicación de las actas de las X JICS y, por ello, cuenta con un tamaño mayor del habitual. Esperamos repetir este número especial el próximo año con la undécima edición de las Jornadas de Innovación y Calidad del Software. Agradecemos la labor del comité de programa coordinado por la Dr. M.Idoia Alarcón (Universidad Autónoma de Madrid) y compuesto por la siguiente lista de expertos:

- Antonia Mas (Universitat de les Illes Balears)
- Luis de Salvador (AGPD)
- Ricardo Vargas (Universidad del Valle de Méjico)
- Javier Tuya (Universidad de Oviedo)
- Antonio de Amescua (Universidad Carlos III de Madrid)
- María Moreno (Universidad de Salamanca)
- José Antonio Calvo-Manzano (Universidad Politécnica de Madrid)
- José Antonio Gutiérrez de Mesa (Universidad de Alcalá)
- Isabel Ramos (Universidad de Sevilla)
- Esperança Amengual (Universitat de les Illes Balears)
- José Ramón Hilera (Universidad de Alcalá)
- Mercedes Ruiz (Universidad de Cádiz)
- María Teresa Villalba (Universidad Europea de Madrid)
- Adolfo Vázquez (INSA)
- María José Escalona (Universidad de Sevilla)
- Ana Araújo (Ministerio de Medio Ambiente)
- Antonio Rodríguez (Telelogic)
- Gurutze Miguel (TQS)
- Beatriz Pérez (Centro de Ensayos de Software, Uruguay)
- José Javier Martínez (Universidad de Alcalá)
- José Díaz (SSQTB)

Luis Fernández Sanz

Hacia la gestión cuantitativa en la gestión de proyectos en el ámbito de las pymes

Jose A. Calvo-Manzano
Facultad de Informática. Universidad Politécnica de Madrid (UPM)
jacalvo@fi.upm.es
Iván García, Magdalena Arcilla
Universidad Tecnológica de la Mixteca-México. ETS Ingeniería Informática.
Universidad Nacional de Educación a Distancia (UNED)
ivan@mixteco.utm.mx; marcilla@issi.uned.es

Abstract

Project management is a key factor for software project success. Moreover, taking into account that small and medium enterprises are the 99.87% of the Spanish enterprises, it is vital for this type of enterprises to implement the processes involved in project management. Although there are process models and tools that cover project management, however these models and tools are oriented to large enterprises. So, it is necessary to adapt these models and tools to the small and medium enterprises. A complementary solution based on the implementation of a process asset library is presented. This library will allow establishing a project management standard process to be tailored to the enterprise's projects.

Key words: project management, process asset library, measurement repository, CMMI-DEV.

Resumen

La gestión de proyectos es un aspecto clave en el éxito de los proyectos de software. Si, además, se tiene en cuenta que las pequeñas y medianas empresas en España constituyen el 99,87% de las empresas españolas, es evidente que resulta crucial para estas empresas, y para la economía nacional, una buena implantación de los procesos involucrados en la gestión de proyectos. Aunque existen modelos de procesos y herramientas que cubren los aspectos de gestión de proyectos, sin embargo estos modelos y herramientas se han orientado a las grandes empresas, lo que hace necesaria su adaptación a las condiciones particulares de las pequeñas y medianas empresas. Se presenta una solución complementaria para este tipo de empresas basada en la implantación de un repositorio de activos de proceso que permita establecer un proceso estándar de gestión de proyectos y su adaptación a las condiciones específicas de cada proyecto de la empresa.

Palabras clave: gestión de proyectos, repositorio de activos, repositorio de medición, CMMI-DEV.

1. Introducción

La llamada "crisis del software" del año 1969 perdura hasta nuestros días, hasta el punto de que aún están presentes los problemas relativos al fracaso de los proyectos. De acuerdo con Jones [1] y el Standish Group International [2]:

- El 30% de los proyectos de software son cancelados.
- El 50% de los proyectos se abandonan o exceden los costes previstos.
- El software falla a menudo (en un 60%), dada su baja calidad.
- La entrega del software es tardía en 9 de cada 10 proyectos.

El Standish Group [3] y Dove [4] confirman la carencia de gestión como una de las causas del fracaso.

En el mundo se realiza un millón de proyectos cada año. Cairó [5] indica que aproximadamente un tercio de estos proyectos se extralimita un 125% en tiempo y coste. Pero, ¿a qué se deben tantos fracasos? En el mismo estudio, Cairó indica que una de las razones más importantes es la gestión del proyecto.

Particularizando el problema, Jones [6] ha identificado tres grandes causas de los retrasos y fracasos en los proyectos de una organización: una estimación inexacta, la pobre comunicación del estado del proyecto y la falta de información histórica. Dichas causas son aspectos claves de la planificación del proyecto, y de su seguimiento y control. Jones también precisa que cada una de estas causas puede eliminarse con una adecuada gestión de los proyectos.

En un estudio del Departamento de Defensa de EEUU sobre proyectos de software se indica [7]: "después de dos décadas de promesas incumplidas sobre el aumento de la productividad y la calidad con el uso de nuevas metodologías y tecnologías, las industrias y las organizaciones gubernamentales se han dado cuenta de que el problema fundamental de los proyectos de software es la incapacidad de gestionar el proceso de software, en especial la pobre gestión de los riesgos".

Para tratar estos problemas, la mayoría de las investigaciones se ha centrado solo y exclusivamente en el aspecto tecnológico de la gestión de proyectos, desarrollando aplicaciones como Primavera P6® [8], Open Plan Professional [9], Microsoft Project [10], PMLP [11] y EPM [12], que cubren únicamente elementos de la planificación y el

seguimiento de los proyectos de software. Tales iniciativas incluyen los procesos que tratan las actividades relativas al:

- Establecimiento y mantenimiento del plan del proyecto (proceso de planificación).
- Establecimiento y mantenimiento de compromisos (procesos de planificación, y de seguimiento y control).
- Supervisión del progreso frente al plan y toma de acciones correctivas (proceso de seguimiento y control).
- Gestión del riesgo (proceso de riesgos).

Sin embargo, estas iniciativas no han tenido en cuenta aspectos tales como:

- Establecimiento y mantenimiento de un conjunto válido de activos de procesos de gestión de proyectos de la organización: procesos estándares de gestión de proyectos de la organización, descripciones de modelos de ciclo de vida, pautas y criterios de adaptación de los procesos estándares de gestión de proyectos de la organización, catálogo de medidas de la organización, repositorio de activos y lecciones aprendidas (proceso de definición de los procesos de gestión de proyectos según CMMI-DEV [13]).
- Establecimiento de los procesos de gestión de proyectos para cada proyecto, los cuales se adaptan a partir del conjunto de procesos estándares de gestión de proyectos de la organización (proceso de gestión integrada del proyecto según CMMI-DEV [13], incluyendo mecanismos de adaptación).
- Planificación, implementación y despliegue de las mejoras de los procesos estándares de gestión de proyectos de la organización, a partir de una comprensión de las fortalezas y debilidades de los procesos de gestión de proyectos y de los activos de proyectos de la organización (proceso de mejora de los procesos de gestión de proyectos según CMMI-DEV [13], teniendo en cuenta los factores humanos para lograr el despliegue a la organización).
- Definición de modelos cuantitativos que permita predecir el comportamiento futuro de los proyectos (en términos de rendimiento de los procesos de gestión de proyectos y de calidad de los productos) basándose en información histórica de la organización y proporcionando esta forma de información para la mejora (proceso

de gestión cuantitativa de los procesos de gestión de proyectos según CMMI-DEV [13]).

Así pues, resulta conveniente proponer una solución complementaria que contemple estos procesos.

A partir de comienzos de los años noventa, la industria y los investigadores interesados en ingeniería del software han expresado un interés especial en la mejora del proceso de software (SPI, *Software Process Improvement*) [14]. Una muestra de ello es la aparición de iniciativas internacionales relacionadas con SPI, tales como CMMI-DEV [13], ISO/IEC 15504:2004 [15] (anteriormente conocido como SPICE) e ISO/IEC 12207:2004 [16].

Asimismo, se han desarrollado diferentes métodos para evaluar el estado actual de los procesos de una organización, como SCAMPI (*Standard CMMI Appraisal Method for Process Improvement*) [17], ISO/IEC 15504. Partes 2 y 3 [15] y *CMM Based Appraisal for Internal Process Improvement* [18], y modelos de mejora como IDEAL [19] e ISO/IEC 15504. Parte 4 [15]. Sin embargo, no se ha trabajado en el desarrollo de métodos que guíen en el desarrollo y la implantación de mejoras prácticas, de tal modo que actualmente los costes de implantación son muy elevados para cualquier tipo de empresa, costes difícilmente abordables por las pequeñas y medianas empresas. La implantación de un proceso efectivo de software en cualquier empresa requiere inversión de tiempo y dinero [20], [21] y [22].

Esta inquietud que se inició en las grandes compañías se ha trasladado recientemente a las pequeñas empresas. En enero del 2006 existían en España alrededor de tres millones de pymes, que representan el 99,87% del total de empresas. Este dato revela la importancia de las pymes a nivel macroeconómico (véase Tabla 1).

Empresas	Microempresas (incluye autónomos, 0-9)	Pequeñas (10-49)	Medianas (50-(249)	Pymes (0-249)	Grandes (250 y más)	Total
Nº asalariados	2.642.775	145.418	21.192	2.809.385	3.735	2.813.120
Porcentaje total	93,94%	5,17%	0,75%	99,87%	0,13%	100%

Tabla 1. Empresas españolas según estrato de asalariados y porcentaje del total (DIRCE 2007).

Sin embargo, debido a que los modelos han sido orientados a las grandes empresas y a que, desafortunadamente, muy pocos estudios [24] [25] han centrado su interés en el uso de las prácticas efectivas hacia las características de las pymes [23], el conocimiento de los modelos por parte de estas empresas ha sido débil. Incluso si una empresa pequeña los conoce y reconoce las necesidades de mejorar sus procesos, sus recursos (financieros y de personal) son limitados [26]. Kuvaja [27], Kilpi [28] y Peirano [29] determinan como principal obstáculo la cantidad de dinero que las pymes pueden invertir al intentar adoptar un modelo o tecnología nueva. Según Weigers [30], el mayor error en la implantación de programas de mejora en pymes se da por la falta de seguimiento de los planes de implantación, ya que estas actividades son muy costosas de realizar, por consumir mucho tiempo y recursos.

Así pues, es necesaria la adaptación de los modelos existentes a las condiciones particulares de las pequeñas empresas. Teniendo esto en cuenta, el objeto del presente artículo se dirige a presentar una aproximación a los procesos de gestión de proyectos en las pymes desarrolladoras de software, sobre la base de costes abordables por ellas.

2. El proceso de gestión de proyectos

El proceso de gestión de proyectos cubre las actividades de gestión del proyecto relativas a la planificación, seguimiento y control del proyecto. De acuerdo con el CMMI-DEV v1.2 [13], este proceso engloba las áreas de proceso de: Planificación del Proyecto (*Project Planning*, PP), Seguimiento y Control del Proyecto (*Project Monitoring and Control*, PMC), Gestión Integrada del Proyecto (*Integrated Project Management*, IPM), Gestión de riesgos (*Risk Management*, RSKM) y Gestión Cuantitativa del Proyecto (*Quantitative Project Management*, QPM).

Los procesos básicos de la gestión de proyectos (planificación, seguimiento y control) se encargan de las actividades relativas al establecimiento y mantenimiento del plan del proyecto, establecimiento y mantenimiento de compromisos, seguimiento del progreso frente al plan y toma de acciones correctivas (véase Figura 1).

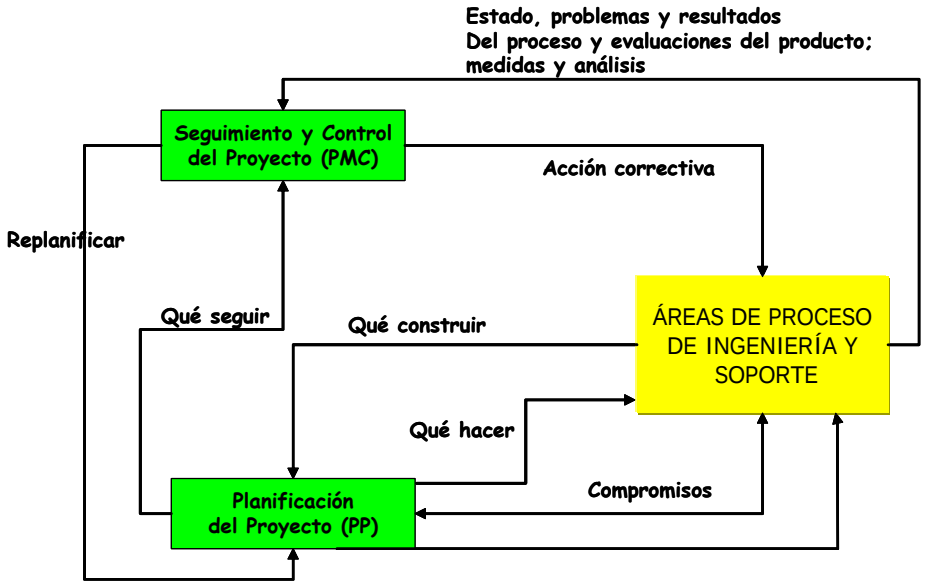


Figura 1. Procesos básicos de la gestión de proyectos.

Los procesos avanzados de la gestión de proyectos (gestión integrada del proyecto, gestión de riesgos y gestión cuantitativa del proyecto) tratan actividades como el establecimiento de un proceso definido que se adapta a partir del conjunto de procesos estándares de la organización, la coordinación y colaboración con las partes interesadas relevantes, la gestión de los riesgos y la gestión cuantitativa del proceso definido del proyecto.

3. Hacia los niveles de capacidad gestionado, definido y gestionado cuantitativamente

A continuación se presenta una definición básica de lo que significa alcanzar un determinado nivel de capacidad en la gestión de proyectos.

- Alcanzar el nivel gestionado en la gestión de proyectos significa que todos los proyectos de la organización utilizan un proceso de gestión de proyectos. Es decir, cada proyecto de la organización utiliza su propio proceso de gestión de proyectos, el cual, en la mayoría de los casos, solo es conocido por el respectivo jefe de proyecto. En definitiva, los estándares, descripciones de proceso y procedimientos son diferentes en cada proyecto.

- Alcanzar el nivel definido en la gestión de proyectos significa que todos los proyectos de la organización utilizan el proceso estándar de gestión de proyectos de la organización o un proceso adaptado de dicho proceso estándar. Es decir, todos los proyectos de la organización utilizan el mismo proceso de gestión de proyectos o una adaptación de este (denominado proceso definido en terminología de CMMI-DEV). Esto significa que el proceso de gestión de proyectos utilizado es conocido por toda la organización, a diferencia del nivel gestionado. En este nivel la organización dispone de un repositorio/biblioteca de activos de proceso que almacena los diferentes procesos estándares que la organización utiliza en sus proyectos, los ciclos de vida, los criterios para adaptar los procesos estándares a los diferentes proyectos y el repositorio de medición (donde se almacenan las medidas y métricas que se toman de los diferentes procesos, tareas y productos).
  Una vez que una empresa (grande o pyme) ha alcanzado el nivel 2 de capacidad (por ejemplo, en el proceso de gestión de proyectos), alcanzar el nivel 3 de capacidad es más sencillo para una pequeña y mediana empresa, ya que en estas empresas es más fácil que todo el personal de la empresa (en el 93,94% de las empresas, no más de 10 personas, de acuerdo con la Tabla 1) utilice el mismo proceso de gestión de proyectos o una versión adaptada de dicho proceso.
- Alcanzar el nivel gestionado cuantitativamente significa que se aplican técnicas estadísticas y cuantitativas para gestionar el rendimiento del proceso de gestión de proyectos y la calidad del producto. Una vez establecido el repositorio de medición, sería necesario aplicar análisis estadísticos sobre la información de procesos/productos almacenada en dicho repositorio.

4. La biblioteca de activos de proceso

Para llegar al nivel de capacidad 3 se necesita disponer de una biblioteca/repositorio de activos de proceso bien estructurada e implementada. Las experiencias de implementación de la mayor parte de las organizaciones que están en niveles de madurez/capacidad avanzados establecen que cuentan con una biblioteca de activos de proceso bien organizada e implementada, y que esta es la llave que habilita a las organizaciones para tener una cultura enfocada a la madurez de los procesos [23].

Una biblioteca de activos de proceso proporciona el conocimiento esencial para obtener, definir y diseminar los procesos de una organización; además, es el instrumento fundamental para compartir este conocimiento a través de toda la organización [31].

La biblioteca de activos de proceso (véase Figura 2) es un repositorio común de información (activos) que permitirá la estandarización de los procesos, en el sentido de que todos los proyectos utilizarán el mismo proceso estándar (ciclo de vida) o distintas adaptaciones aprobadas del proceso estándar.

Las métricas y medidas constituyen el repositorio de medición de la organización, el cual contendrá datos a nivel de proceso, producto y tarea (planificados y reales) de los diferentes proyectos que se vayan realizando en la empresa.

Cada proyecto seleccionará su propio proceso (proceso definido de gestión de proyectos) según las guías y los criterios de adaptación (que hemos denominado "patrones"). Este proceso definido será utilizado por los proyectos que realimentarán la propia biblioteca de activos a dos niveles: con las medidas reales a nivel de proceso, tarea y producto, y con las lecciones aprendidas y mejoras correspondientes.

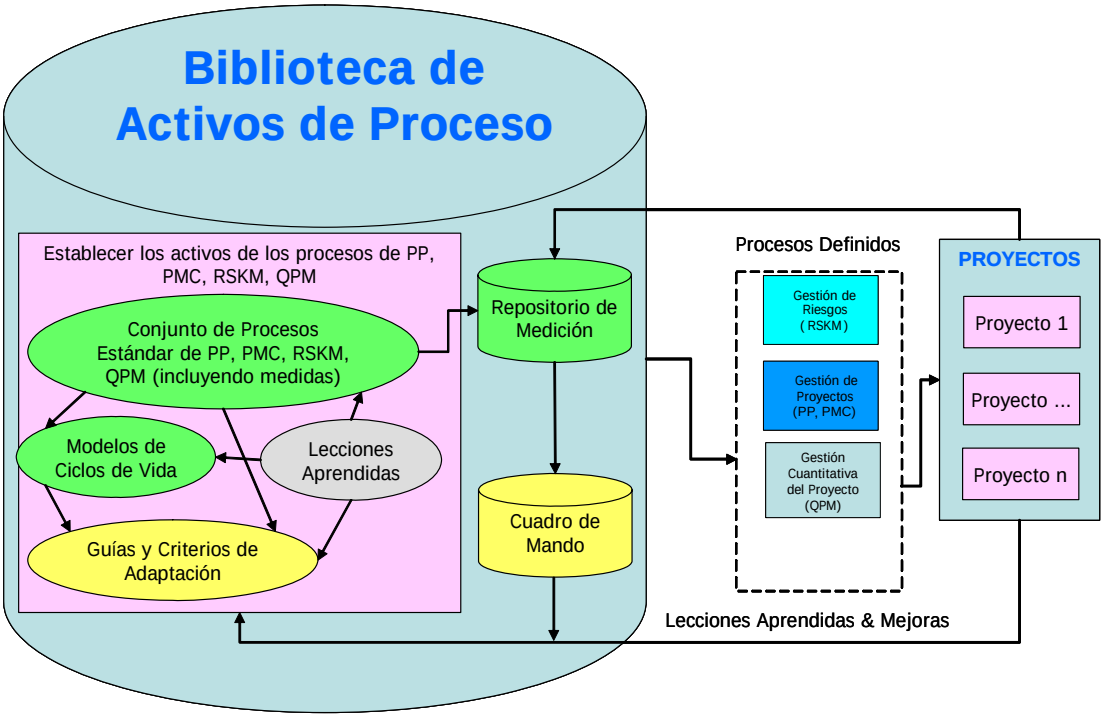


Figura 2. Biblioteca de activos de gestión de proyectos.

En el caso de la gestión de proyectos, en el repositorio de medidas se almacenarán, por ejemplo, los valores estimados y reales de los esfuerzos relativos a las diferentes tareas de los proyectos. Si se tiene en cuenta que los proyectos siguen un proceso estándar definido (es decir, diferentes proyectos aplicarán las mismas tareas), a través del análisis estadístico se podrán estimar los rendimientos de futuros proyectos, que serán calculados en función del histórico de datos.

Cada proceso estándar de la biblioteca de activos se ha definido en términos de tareas, productos, métricas (de proceso, tareas y productos) y activos (de proceso, tareas, productos y métricas), según se refleja en la Figura 3. Estos procesos (estándares de la organización) serán usados por los proyectos de la organización, teniendo en cuenta las guías y criterios de adaptación (los patrones).

Los activos son artefactos o mecanismos que facilitarán el soporte para realizar los procesos, tareas, productos y métricas. Los patrones son los diferentes procesos definidos que una organización puede llevar a cabo. Las métricas estándares serán indicadores para la toma de decisiones por parte del negocio.

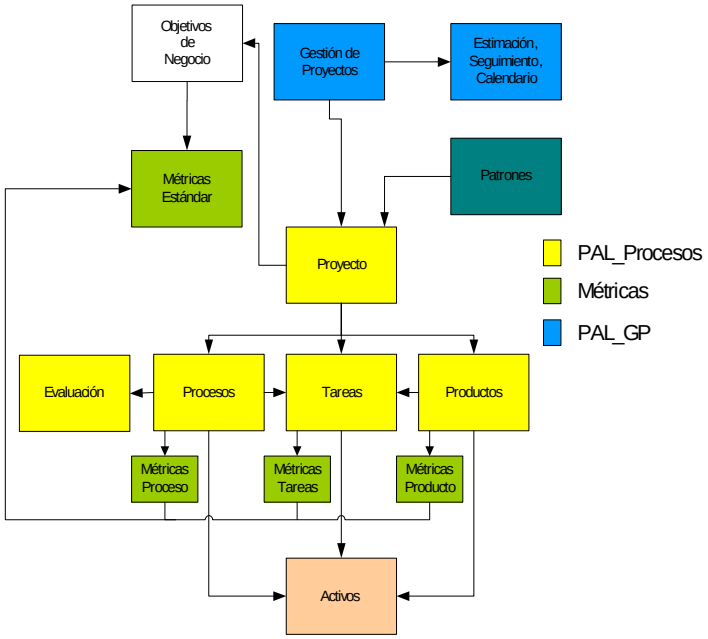


Figura 3. Detalle de la biblioteca de activos.

La Figura 4 muestra la pantalla principal del repositorio de activos que se ha definido; en ella se aprecian cuatro partes bien diferenciadas: definición de estándares de proceso y gestión de activos, definición del repositorio de medición, definición de patrones y gestión de proyectos. Si se añaden nuevos procesos al repositorio, puede ser necesario añadir nuevos conceptos al repositorio; así, por ejemplo, en el caso del proceso de gestión de configuración, se añade, entre otros, el concepto de línea base.

Figura 4. Menú principal de la biblioteca de activos.

5. Conclusiones

La PAL implementada cuenta con activos para los procesos de Planificación de Proyectos (PP), Seguimiento y Control de Proyectos (PMC), y Gestión de Requisitos (REQM). Estos activos permitirán implementar las actividades o tareas necesarias para cumplir los objetivos del proceso correspondiente. La PAL se ha probado experimentalmente en proyectos con alumnos de la Facultad de Informática de la Universidad Politécnica de Madrid y en la actualidad está proyectada su validación en un proyecto piloto.

Como trabajo futuro, se están desarrollando los activos correspondientes a otros procesos de CMMI-DEV (concretamente, PPQA), así como la incorporación de otros modelos de proceso y/o metodologías, como CMMI-ACQ, ITIL (inicialmente, el proceso de Gestión de Nivel de Servicio), *Team Software Process* y UML.

Asimismo, se pretende que los patrones se realicen de forma automática, por ejemplo, en función de una serie de criterios determinados, como el tamaño del proyecto (pequeño, mediano o grande) y el sector del proyecto (automoción, defensa).

Referencias

[1] Jones, G., *Software Engineering*, John Wiley & Sons, 1990.

[2] Standish Group International, *2006 Third Quarter Research Report*, The Standish Group International, 2006.

[3] Standish Group International, *Extreme Chaos*, The Standish Group International, 2001.

[4] Dove, R., *Value Propositioning. Book One. Perception and Misperception in Decision Making*, Iceni Books, 2004.

[5] Cairó, O., *Proyecto KAMET II*, Instituto Tecnológico Autónomo de México, 2004.

[6] Jones, C., "Why Flawed Software Projects Are Not Cancelled in Time", *Cutter IT Journal*, vol. 16, nº 12, pp. 12-17, 2003.

[7] Brock, S., Hendricks, D., Linnell, S. y Smith, D., *A Balanced Approach to IT Project Management*, ACM Publications. Proceedings of SAICSIT, 2003.

[8] Primavera Systems, *Primavera P6®*, http://www.primavera.com/products, julio 2008.

[9] Deltek, *Deltek Open Plan*, http://www.welcom.com/products/evp/default.asp, julio 2008.

[10] Microsoft, *Microsoft Office Project 2007*, http://office.microsoft.com/es-es/project/FX100487773082.aspx, julio 2008.

[11] Hussain, S., *PMLP-Project Management Using Temporal Logic Programming*, IEEE Software, 2000.

[12] Bull España, *EPM como garantía de aplicación de una metodología de gestión de proyectos en una corporación*, Soluciones empresariales. Microsoft, 2002.

[13] Team, C. P., *Capability Maturity Model Integration for Development (CMMI-DEV) version 1.2, CMU/SEI-2006-TR-008*, Software Engineering Institute. Carnegie Mellon University, 2006.

[14] Pino, F., García, F. y Piattini, M., "Revisión sistemática de procesos software en micros, pequeñas y medianas empresas", *Revista Española de Innovación, Calidad e Ingeniería del Software*, vol. 2, nº 1, pp. 6-23, 2006.

[15] ISO/IEC 155504:2004, *Information Technology-Process Assessment. Parts 1-5*, International Organization for Standardization, 2004.

[16] ISO/IEC 12207:2002/FDAM 2, *Information Technology-Software Life Cycle Processes,* International Organization for Standardization, 2004.

[17] Members of the Assessment Method Integrated Team, *Standard CMMI® Appraisal Method for Process Improvement (SCAMPI), Version 1.1 (CMU/SEI-2001-HB-001),* Software Engineering Institute. Carnegie Mellon University, 2001.

[18] Dunaway, D. K. y Masters, S., *CMM-Based Appraisal for Internal Process Improvement (CBA IPI): Method Description, Technical Report CMU/SEI-96-TR-007,* Software Engineering Institute. Carnegie Mellon University, 1996.

[19] McFeeley, B., *IDEAL^{SM}: A User'2 Guide for Software Process Improvement*. *Handbook CMU/SEI-96-HB-001*. Software Engineering Institute. Carnegie Mellon University, 1996.

[20] Hersleb, J., Carleton, A., Rozum, J., Siegel, J. y Zubrow, D., *Benefits of CMM-Based Software Process Improvement: Initial Results. CMU/SEI-94-TR-013,* Software Engineering Institute. Carnegie Mellon University, 1994.

[21] Hersleb, J. y Goldenson, D., *After the Appraisal: A Systematic Survey of Process Improvement, Its Benefits, and Factors that influence the Success. CMU/SEI-95-TR-009,* Software Engineering Institute. Carnegie Mellon University, 1995.

[22] Mondragón, O., "Addressing Infrastructure Issues in Very Small Settings", *Proceedings of the First International Research Workshop for Process Improvement in Small Settings,* pp. 5-10, 2005.

[23] García, S., Graettinger C. Y Kost, K., *First International Research Workshop for Process Improvement in Small Settings. Special Report CMU/SEI-2006-SR-001,* Software Engineering Institute. Carnegie Mellon University, 2006.

[24] Mas, A. y Amengual, E., "La mejora de los procesos de software en las pequeñas y medianas empresas (pymes). Un nuevo modelo y su aplicación a un caso real", *Revista Española de Innovación, Calidad e Ingeniería del Software, v*ol. 1, nº 2, pp. 7-29, 2005.

[25] Calvo-Manzano, J. A., Cuevas, G., San Feliu, T., Amescua, A. y Pérez, M., "Experiences in the Application of Software Process Improvement in SMES", *Software Quality Journal*, vol. 10, nº 3, pp. 261-273, 2002.

[26] Dyba, T., "Factors of Software Process Improvement Success in Small and Large Organizations: An Empirical Study in the Scandinavian Context", *Proceedings of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pp. 148-157, 2003.

[27] Kuvaja, P. y Messnarz, R., "BootStrap. A Modern Software Process Assessment and Improvement Methodology", *Proceedings of the Fifth European Conference on Software Quality*, 1996.

[28] Kilpi, T., "Product Management Challenge to Software Change Process: Preliminary Results from Three SMEs Experiment", *Software Process: Improvement and Practice*, vol. 3, nº 3, pp. 194-207, 1997.

[29] Peirano, F. y Suárez, D., "Las TICS mejoran el desempeño de las pymes. ¿Somos capaces de explicar cómo lo hacen?", *Simposio sobre la Sociedad de la Información (SSI 2005). Rosario (Argentina), septiembre de 2005.*

[30] Weigers, K. E. y Sturzenberger, D. C., "A Modular Software Process Mini-Assessment Method", *IEEE Software*, vol. 17, n° 1, pp. 62-69, 2000.

[31] Groarke, B., "Web-Based Software Process Improvement Repository", *CrossTalk The Journal of Defense Software Engineering*, vol. 13, nº 3, pp. 24-25, 2000.

Problemas de las pymes en el nivel 2 de madurez. Una muestra sesgada

Juan José Cukier
Pragma Consultores
jcukier@pragmaconsultores.com

Abstract

This study quantifies the experience gathered in 27 SCAMPI class A, B and C appraisals for CMMI© Maturity Level 2, performed between June 2006 and June 2008. All these appraisals were performed on Small and Medium Enterprises (SMEs) with the objective of identifying CMMI components with a high concentration of weaknesses. These weaknesses may put the results of an official SCAMPI[SM] class A appraisal at risk and, hence, require special attention in a software process improvement project.

Key words: CMMI, SCAMPI, pyme, Maturity Level 2.

Resumen

Este estudio cuantifica la experiencia recogida en 27 evaluaciones SCAMPI clases A, B y C del Nivel 2 de Madurez de CMMI en el período junio 2006-junio 2008, realizadas sobre pequeñas y medianas empresas (pymes). Se identifican componentes del modelo CMMI donde se concentran debilidades que más frecuentemente pueden poner en riesgo la implantación exitosa del modelo y la obtención de una acreditación oficial. Estos componentes requieren atención especial durante el proyecto de mejora de procesos.

Palabras clave: CMMI, SCAMPI, pyme, nivel 2 de madurez.

1. Introducción

1.1. Objetivos del estudio

En este estudio se busca cuantificar la experiencia recogida en 27 evaluaciones SCAMPI clases A, B y C del nivel 2 de madurez de CMMI durante el período de tiempo comprendido entre junio de 2006 y junio de 2008. Estas evaluaciones se llevaron a cabo sobre pequeñas y medianas empresas (pymes), identificando componentes de CMMI donde se concentran debilidades que con más frecuencia pueden poner en peligro la implantación exitosa del modelo. Una debilidad es, según la definición del SEI, “la implementación no efectiva, o falta de implementación, de una o más prácticas de CMMI” [1].

1.2. El modelo CMMI

Capability Maturity Model Integration o CMMI es un modelo de procesos basado en las mejores prácticas de la industria.

Las dimensiones críticas de una organización son el personal, los procesos y las herramientas. Y de estas, los procesos son los encargados de unir las otras dos con el propósito de alcanzar los objetivos del negocio. El foco en los procesos ayuda a construir una plataforma de mejora continua, al trascender estos en el tiempo con menor obsolescencia que la tecnología, y menor rotación que el personal.

El *Software Engineering Institute* (SEI) de la Universidad Carnegie Mellon de EEUU ha creado CMMI bajo la premisa de que la "calidad de un producto o servicio está altamente influenciada por la calidad de los procesos que los producen y los mantienen" [2]. Así, la mejora continua de los procesos podría incrementar paulatinamente el nivel de capacidad y madurez de una organización.

1.3. Evaluaciones CMMI

Una evaluación de CMMI corresponde al estudio y análisis de uno o más procesos realizado por un equipo capacitado de profesionales, utilizando un modelo de referencia de evaluación como base para determinar fortalezas y debilidades dentro de una organización de software [3].

El SEI ha publicado dos documentos guías para realizar una evaluación de CMMI:

- *Appraisal Requirements for CMMI* (ARC).
- *Standard CMMI Appraisal Method for Process Improvement* (SCAMPI).

ARC define un conjunto de requerimientos considerados esenciales para realizar una evaluación CMMI, mientras que SCAMPI es el documento de referencia para la evaluación. Se definen en ARC tres clases de evaluaciones: clase A, clase B y clase C. Las clases definen los requerimientos que debe cumplir la evaluación y determinan tanto la posibilidad de adaptación del método como la profundidad de la evaluación.

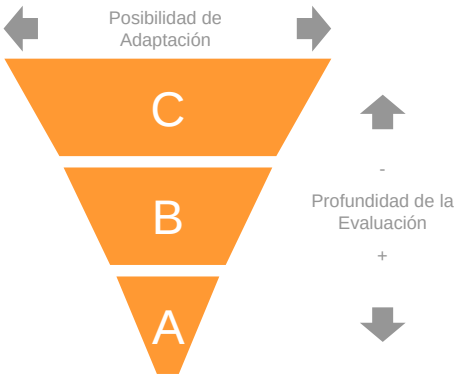


Figura 1. Clases de método SCAMPI.

SCAMPI clase A es el único que otorga acreditaciones oficiales, las cuales deben ser lideradas por un evaluador líder autorizado por el SEI.

SCAMPI clase B está basado en la evaluación clase A. Generalmente se ejecuta cuando la organización necesita evaluar sus procesos con miras a una acreditación oficial. Menos formal aún, de menor duración y con menos información requerida es el SCAMPI clase C.

El método SCAMPI, independientemente de su clase, caracteriza componentes del modelo. Es decir, asigna un puntaje según reglas que varían por clase. De acuerdo con la evidencia objetiva, se realizan transformaciones de datos donde se identifica el cumplimiento (o no) de las prácticas del modelo. Esto genera debilidades, que son agregadas y resumidas en este trabajo en busca de tendencias generales.

1.4. CMMI y pymes

Desde el surgimiento de CMMI se ha cuestionado su aplicación a las pequeñas y medianas empresas (pymes). Incluso la definición de pyme ha ido cambiando a medida que organizaciones más y más pequeñas comenzaron a adoptar exitosamente el modelo. Hoy el SEI define como pyme a aquella organización de 100 personas o menos [4].

En general, las pymes tienen recursos limitados y su éxito suele residir en la excelencia en algún nicho de mercado. Sin embargo, presiones de mercado e incentivos impositivos de varios países, que ven la industria de software como una fuente importante de divisas, han llevado a muchas pymes a evaluar seriamente la implantación de CMMI

como forma de demostrar madurez en sus procesos de desarrollo y mantenimiento de software.

Esta tendencia puede verse reflejada en la Figura 2, que recoge información sobre los reportes de madurez publicados semestralmente por el SEI [4]. El gráfico destaca la composición porcentual de las organizaciones del mundo, según su tamaño, que han adoptado CMMI.

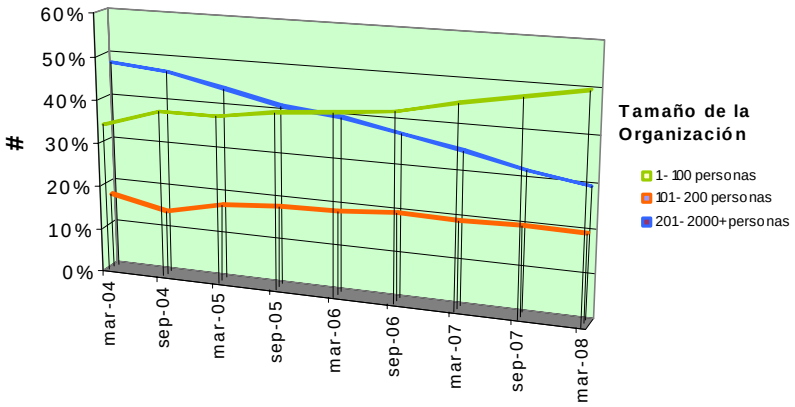


Figura 2. Adopción de CMMI según el tamaño de la organización.

Significativamente, las organizaciones que tienen entre 1 y 100 empleados (línea verde), a partir de fines del año 2005 se convirtieron en la primera minoría. Es decir, hay más pymes que han adoptado CMMI que ningún otro tipo de organización, atendiendo al tamaño. Y según el último reporte de madurez, de marzo de 2008 [4], las pymes representan casi el 50% de todos los niveles de madurez reportados en el mundo.

Si bien estas cifras son contundentes y subrayan la factibilidad técnica y económica de las pymes para la implantación de CMMI, este tipo de organizaciones suelen reflejar algunos problemas recurrentes a la hora de adoptar el modelo. Este estudio, realizado a través de 27 SCAMPI clase C y clase A en 18 pymes busca identificar estos problemas, permitiendo así un tratamiento especial durante los proyectos de mejora de procesos.

1.5. El nivel 2 de madurez

El foco del nivel 2 de madurez está en la gestión de proyectos. En la Tabla 1 se detallan las áreas de proceso del nivel 2. Como se puede observar, la única área de proceso de

ingeniería del nivel 2 es la gestión de requisitos; el resto se presenta en el nivel 3. Una interpretación posible de esta decisión de diseño reside en la necesidad de poder gestionar un proyecto adecuadamente, de principio a fin, para mejorar los aspectos más técnicos del desarrollo y el mantenimiento de sistemas.

Abreviatura	Área de proceso	Propósito
PP	*Project Planning*	Establecer y mantener planes que definen las actividades de los proyectos.
PMC	*Project Monitoring and Control*	Facilitar un entendimiento del progreso del proyecto.
REQM	*Requirements Management*	Gestionar los requerimientos del producto y sus componentes.
CM	*Configuration Management*	Establecer y mantener la integridad de los productos.
MA	*Measurement and Analysis*	Desarrollar una capacidad de medición para dar soporte a la gestión.
PPQA	*Process and Product Quality Assurance*	Proveer una visión objetiva de los procesos y los productos asociados.

Tabla 1. Áreas de proceso del nivel 2 de madurez, excluyendo SAM (*Supplier Agreement Management).*

La Figura 3 muestra esquemáticamente el desafío de la transición de una organización hacia el nivel 2.

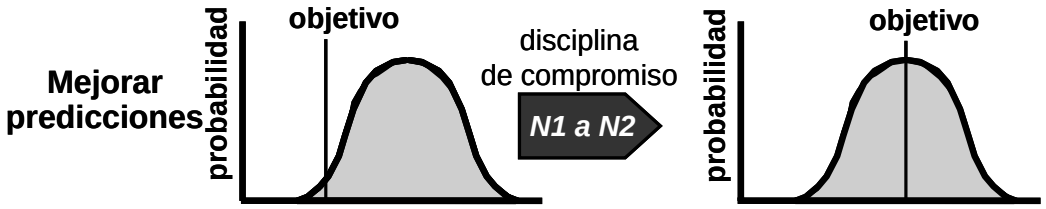


Figura 3. Evolución del nivel 1 al nivel 2 de madurez.

En el nivel 2 de madurez se genera una disciplina de compromiso, se mejoran las predicciones y los planes, pudiéndose establecer, así, compromisos más razonables. Se disparan acciones correctivas ante problemas, que se siguen luego hasta su cierre. Los riesgos se atacan más proactivamente. Los plazos de ejecución pueden parecer mayores a lo experimentado antes de la implantación del modelo, pero parte de esta diferencia puede explicarse en una "sinceridad" de la organización respecto a su *performance,* tiempos y costos.

2. La muestra

La muestra incluyó un total de 27 evaluaciones SCAMPI de clases A, B y C, del nivel 2 de madurez. Las evaluaciones se ejecutaron entre junio de 2006 y junio de 2008, en 18 pymes de Argentina, Chile y España.

Esta muestra no busca ser significativa desde un punto de vista estadístico. Y contiene, indefectiblemente, un desvío inducido por el líder de la evaluación, que lideró o participó de forma activa en cada una de ellas. La formación académica, la experiencia profesional y personal, y la cultura de origen, entre otros, forman nichos de foco difíciles de erradicar durante la evaluación.

No obstante, esta muestra sí pretende resumir la experiencia personal y los problemas comunes hallados, así como cuantificar los resultados observados día a día.

3. Análisis de los datos

A partir del análisis cuantitativo de los resultados de los SCAMPI se intentó responder a las siguientes preguntas:

1. ¿Qué porcentaje de prácticas tienen debilidades, en promedio, en un SCAMPI? ¿Esto varía según la clase de SCAMPI?
2. ¿Cuáles suelen ser las áreas de procesos con más debilidades?
3. ¿Cuáles suelen ser las prácticas específicas con más debilidades?
4. ¿Cuáles suelen ser las prácticas genéricas con más debilidades?

3.1. Porcentaje promedio de debilidades

La Figura 4 indica la distribución porcentual de las 108 prácticas del nivel 2 de madurez, agregando los resultados de todas las clases de SCAMPI. El color verde indica que la práctica no presentó debilidades. El amarillo, que existen debilidades que ponen en riesgo la práctica. Y el rojo, que la práctica no ha sido implementada o que se ha implementado incorrectamente. Como puede observarse, un 80% de las prácticas del nivel 2, en promedio, no registraron debilidades en las evaluaciones, mientras que un 20% de las prácticas de ese nivel de madurez reportaron debilidades leves (color amarillo) o críticas (color rojo).

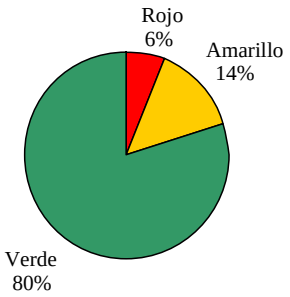


Figura 4. Distribución de caracterizaciones agregada para SCAMPI, clases A, B y C.

Esta composición varía significativamente al segmentar la muestra por clase de SCAMPI. La Figura 5 muestra la distribución para SCAMPI clases C y B *versus* SCAMPI clase A.

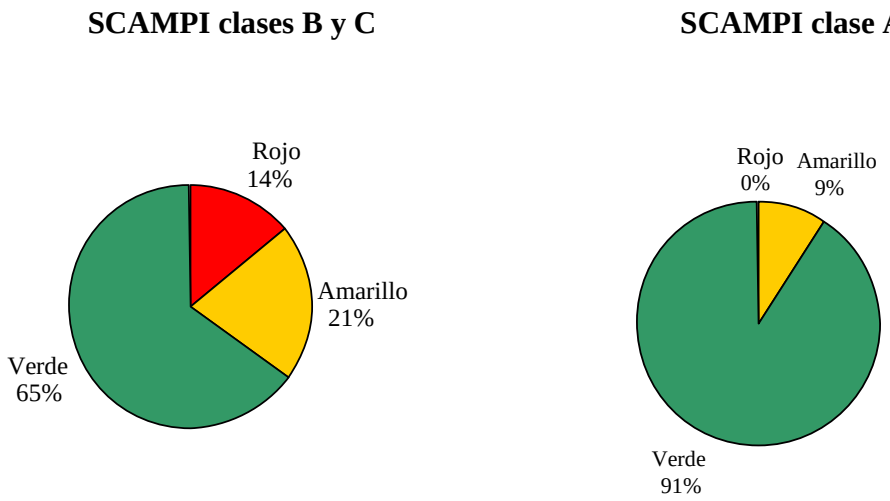


Figura 5. Distribución de caracterizaciones en SCAMPI clases C y B *versus* SCAMPI clase A.

Los cambios en la composición de prácticas con debilidades según la clase de SCAMPI se deben, principalmente, a los diferentes usos de SCAMPI clase A y clases B y C. Los SCAMPI clase B y C, en todos los casos, fueron realizados como forma de evaluación previa con el fin de reforzar conceptos e identificar áreas de potenciales problemas. De allí surgen planes de mejora para hacer frente a estos problemas de manera previa a una evaluación oficial. Es natural, entonces, que haya prácticas en rojo, ya sea

porque no han sido implementadas todavía, o porque se ha implementado parcial o erróneamente.

La ausencia de prácticas en rojo en SCAMPI clase A puede atribuirse, en parte, a dos factores:

1. La ejecución de un SCAMPI clase C o B previo, que alertó sobre los componentes en rojo, generando un plan de mitigación.
2. La ejecución de un *Readiness Review* como parte del SCAMPI clase A, una actividad propia de las evaluaciones clase A que tiende a determinar si el equipo de evaluación y la organización se encuentran en condiciones de conducir la evaluación de la manera en que fue planificada.

Estas actividades, que se tradujeron en controles sobre los riesgos inherentes a la implementación de las prácticas, limitaron, por lo tanto, la existencia de prácticas no implementadas (color rojo) al momento de la evaluación oficial.

3.2. Áreas de proceso con más debilidades

En la Figura 6 se recoge la distribución del total de debilidades encontradas en los 27 SCAMPI, agrupadas por áreas de proceso.

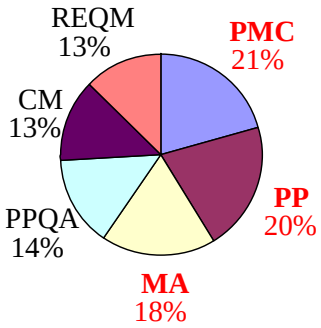


Figura 6. Porcentaje promedio de debilidades por área de proceso.

Este resultado puede explicarse, en parte, entendiendo la distribución de las prácticas dentro del nivel 2 de madurez.

	PP	PMC	MA	PPQA	REQM	CM	Total
Prácticas Específicas	14	10	8	4	5	7	48
Prácticas genéricas	10	10	10	10	10	10	60
Total	**24**	**20**	**18**	**14**	**15**	**17**	**108**

Tabla 2. Prácticas específicas y genéricas por área de proceso.

PP y PMC son las áreas de proceso más voluminosas del nivel 2 de madurez, de modo que entre ambas concentran 44 de las 108 prácticas del nivel (un 40%). Así pues, es razonable esperar que, a mayor concentración de componentes del modelo, exista mayor concentración de debilidades. A la vez, estas áreas de proceso introdujeron conceptos de práctica poco habitual, fuente de potenciales debilidades. Estos son algunos ejemplos:

- Revisar otros planes que puedan afectar al proyecto.
- Reconciliar el trabajo y el nivel de recursos.
- Controlar el compromiso de actores relevantes.
- Controlar la gestión de datos.

El caso de MA fue diferente. La concentración de debilidades se debió a la falta de experiencia de organizaciones inmaduras para la recolección y análisis de mediciones. En general, resultaron ser las prácticas implantadas más recientemente, con las consecuencias inherentes en su institucionalización y su reflejo, por ende, en las debilidades.

La menor concentración de debilidades en CM y REQM se debió a que típicamente son áreas de proceso que las organizaciones tenían implementadas incluso antes de abordar su proyecto de mejora de procesos. Todas las organizaciones reconocían la necesidad de gestionar la configuración de su código fuente ante la presencia de dos o más programadores y de realizar algún tipo de versionado básico de sus documentos (CM). También reconocían la gestión del alcance del proyecto (REQM) como un componente fundamental para el éxito de este.

Con respecto a PPQA, sin bien no suelen ser actividades implementadas en organizaciones inmaduras, su uso durante el proyecto de mejora de procesos es intenso, pues cumple un papel fundamental. En la mayoría de los casos se usó PPQA como la "fuerza institucionalizadora", guiando a los miembros de la organización hacia el uso de

nuevas plantillas y procesos. Este empleo intensivo del proceso se tradujo en un mayor foco en el despliegue de las prácticas, lo que, a su vez, funcionó como cota para las debilidades.

3.3. Problemas más comunes con las prácticas específicas

La Tabla 3 muestra las 10 prácticas con más debilidades del nivel 2 de madurez.

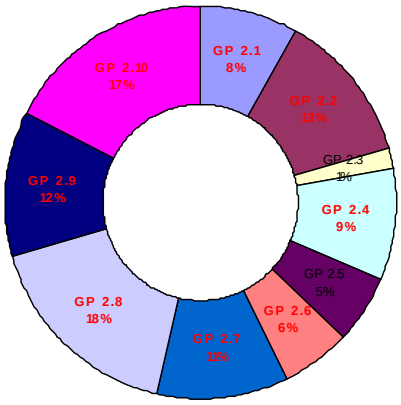
#	Área de proceso	Práctica	Descripción	Problemas más comunes
1	CM	SP 3.2	*Perform configuration audits*	• Líneas base sin auditar y sin criterios de selección muestral. • Auditorías parciales que no aseguran la integridad.
2	MA	SP 1.4	*Specify analysis procedures*	• No se indican parámetros para el análisis. • Criterios no uniformes para en análisis.
3	PMC	SP 1.4	*Monitor data management*	• Contradicciones con lo planificado en PP SP 2.3.
4	PP	SP 3.3	*Obtain plan commitment*	• No se obtiene compromiso externo e interno, o se obtiene pero no se formaliza.
5	PPQA	SP 1.1	*Objectively evaluate processes*	• No hay una evaluación objetiva de todos los procesos relevantes (soporte a GP 2.9). • Los criterios de evaluación son ambiguos, lo que dificulta la objetividad y genera contradicciones en la interpretación por parte de diferentes miembros de la organización.
6	MA	SP 2.2	*Analyze measurement data*	• Métricas en color rojo, indicadores de problemas, pero sin explicaciones de contexto o acciones correctivas.
7	PP	SP 3.1	*Review plans that affect the project*	• Planes *cross* que no han sido considerados en la revisión (pruebas, aseguramiento de la calidad, vacaciones, etc.). • Revisiones realizadas, pero sin rastros de evidencia directa.
8	PP	SP 3.2	*Reconcile work and resource levels*	• Falta de reconciliaciones tras cambios significativos en el proyecto (replanificaciones, rotaciones, cambios de alcance, etc.). • Reconciliaciones realizadas, pero sin rastros de evidencia directa.
9	REQM	SP 1.2	*Obtain commitment to requirements*	• No se evidencia el compromiso interno del equipo de proyecto, con lo que se pone en riesgo la factibilidad técnica.
10	MA	SP 2.4	*Communicate results*	• Las métricas no se comunican con los papeles relevantes, tal como se ha definido. • La comunicación de resultado no asegura una mínima efectividad (por ejemplo, copia a un repositorio común sin informar del evento).

Tabla 3. Prácticas con más debilidades.

3.4. Problemas más comunes con las prácticas genéricas

El propósito de las prácticas genéricas es colaborar con la institucionalización de los procesos. Podríamos decir que si las prácticas específicas se ocupan de los proyectos y la organización, las prácticas genéricas se encargan de las específicas.

La Tabla 4 muestra los problemas más comunes que se han encontrado.

Distribución de debilidades		
Práctica	**Descripción**	**Problemas más comunes**
GP 2.1	*Establish an organizational policy*	• No todos los miembros de los equipos conocen las políticas adecuadas a su papel.
GP 2.2	*Plan the process*	• No siempre existe planificación o calendarización de todos los aspectos relevantes del proceso.
GP 2.4	*Assign responsibility*	• No siempre se explicita al responsable del proceso.
GP 2.6	*Manage configurations*	• No todos los artefactos relevantes del proceso están bajo gestión de configuración.
GP 2.7	*Identify and involve relevant stakeholders*	• No se identifica a todos los actores relevantes, si bien en la práctica se interactúa con ellos. • No siempre hay evidencia del compromiso de los actores relevantes identificados.
GP 2.8	*Monitor and control de process*	• En general, no se monitoriza el proceso según lo planificado (GP2.2).
GP 2.9	*Objectively evaluate adherence*	• No siempre hay aseguramiento de la calidad para todos los procesos o productos de trabajo relevantes.
GP 2.10	*Review status with higher level*	• Las reuniones no suelen estar planificadas ni ser periódicas. • No hay evidencia de la revisión, más allá de la generación de

	management	los informes de estado de los procesos. • No todos los procesos son revisados con la gerencia.

Tabla 4. Problemas más comunes en las prácticas genéricas.

4. Conclusiones

La adopción del nivel 2 de madurez representa un importante desafío para cualquier organización. Las pymes suman a este desafío restricciones presupuestarias, de disponibilidad de recursos humanos y de capacidad de incorporación de nuevos conocimientos.

Este trabajo resume la evidencia empírica de 27 SCAMPI clases A, B y C, destacando las prácticas específicas y genéricas con mayor concentración de debilidades. La distribución de debilidades fue relativamente uniforme entre las prácticas. En CM, sin embargo, puede observarse el principio de Pareto [5], según el cual una pequeña cantidad de prácticas concentra un porcentaje significativo de las debilidades.

Las pymes que implementen el nivel 2 de madurez de CMMI deben considerar que no hay prácticas "seguras". No obstante, una revisión previa a través de un SCAMPI clase C o B con foco en la existencia y adecuación de artefactos directos que evidencien la implementación de las prácticas, parece reducir significativamente los nichos de debilidades y acotar los riesgos inherentes a un SCAMPI clase A.

Futuros trabajos podrán ampliar la base de datos sobre la cual se realizó este estudio, para buscar tendencias segmentando por tipo de empresa, locación geográfica u otras características propias de las organizaciones evaluadas.

Referencias

[1] CMMI Product Team, "MDD Glossary", en: SCAMPI Upgrade Team, *Standard CMMI® Appraisal Method for Process Improvement (SCAMPI^{SM}) A. Version 1.2: Method Definition Document,* Software Engineering Institute. Carnegie Mellon University, 2006.

[2] CMMI Product Team, *CMMI® for Development .Version 1.2*, Software Engineering Institute. Carnegie Mellon University, 2006.

[3] SCAMPI Upgrade Team, *Standard CMMI® Appraisal Method for Process Improvement (SCAMPI^{SM}) A. Version 1.2: Method Definition Document*, Software Engineering Institute. Carnegie Mellon University, 2006.

[4] SEI Customer Relations, *Process Maturity Profile*, Software Engineering Institute. Carnegie Mellon University, 2008.

[5] Pareto, V., *Cours d'economie politique*, Geneva, 1896.

Mejora de procesos organizativos: análisis estadístico

Izaskun Santamaria, Teodora Bozheva, Iñaki Martínez de Marigorta
European Software Institute
{izaskun.santamaria, teodora.bozheva, inaki.marigorta}@esi.es

Abstract

Organisations especially Small and Medium Enterprises aim to improve their competitiveness and their business results. How can they face up to this kind of improvement initiatives? Are software and production processes improvement initiatives enough? This paper outlines some answers to this cornerstone in quality assurance based on stadistical data from ITMark appraisals performed in 44 organisations in Europe and Latin America.

Key words: ITMark, process improvement, business, security, CMMI.

Resumen

Queremos mejorar los procesos organizativos, los resultados del negocio y la competitividad de nuestra empresa. ¿Cómo enfocamos las actividades de mejora? ¿Es suficiente mejorar solo los procesos de desarrollo y producción? Este artículo presenta las respuestas a estas y otras preguntas relacionadas con la mejora de los procesos en organizaciones pequeñas, y lo hace basándose en los datos estadísticos ITMark en 44 organizaciones de Europa y Latinoamérica.

Palabras clave: ITMark, mejora de procesos, negocio, seguridad, CMMI.

1. Los ámbitos de mejora

Para que una pyme alcance un buen posicionamiento en el mercado y rentabilidad en sus desarrollos, debería mejorar diferentes aspectos de su organización y trabajo. Los modelos de calidad que se utilizan hoy en día abarcan algunos de estos aspectos, pero no todos de forma integrada, y tampoco profundizan en las dependencias entre las diferentes líneas de mejora.

El objetivo del método ITMark (http://www.esi.es/index.php?op=15.1.2) es mejorar la eficacia y la competitividad de una pyme a través de la mejora de sus procesos de negocio, de desarrollo y de la seguridad de la información.

Dependiendo de la madurez que demuestra una organización en estos aspectos, se pueden alcanzar niveles ITMark Básico, ITMark Premium e ITMark Elite.

1.1. Procesos de negocio

El modelo de gestión de negocio de ITMark está basado en la experiencia de organizaciones de capital riesgo e incluye diez categorías de procesos de negocio: mercado; dirección; productos y servicios; ventas, *marketing* y distribución; estrategia y comité; análisis financiero; perfil del cliente y análisis; factores de inversión; desarrollo y producción; industria y macroentorno.

En función del perfil de las organizaciones, que se puede definir como embrionario (en concepción), puesta en marcha, desarrollo y expansión, la evaluación define un grado mayor o menor de exigencia.

1.2. Procesos de desarrollo de software y sistemas

La parte de los procesos de desarrollo está basada en el modelo CMMI for Development v. 1.2 0 que se estructura en áreas de proceso. Las que se consideran en el método ITMark son las siguientes (agrupadas por categoría y nivel de madurez):

	Nivel de madurez 2	**Nivel de madurez 3**
Gestión de procesos		OPF: Enfoque a los procesos organizativos OPD: Definición de los procesos organizativos OT: Formación organizativa
Gestión de proyectos	PP: Planificación de proyecto PMC: Seguimiento y control de proyecto SAM: Gestión de los acuerdos con los proveedores	IPM: Gestión integrada de proyectos RSKM: Gestión de riesgos
Ingeniería	REQM: Gestión de requisitos	RD: Desarrollo de requisitos TS: Solución técnica PI. Integración de producto VER: Verificación VAL: Validación
Soporte	CM. Gestión de configuración PPQA: Aseguramiento de la calidad de los procesos y los productos MA: Medición y análisis	DAR: Análisis de decisiones y resolución

Tabla 1. Áreas de proceso CMMI incluidas en ITMark.

1.3. Seguridad de la información

Como ITMark está orientado a pymes, se ha adaptado la serie ISO/IEC 27000 (0, 0) a las necesidades de este tipo de organizaciones. Se han definido tres niveles; los objetivos de cada uno de los niveles son los siguientes:

Nivel	Objetivos
1	Controles lógicos y físicos, indispensables para la organización; procedimientos como los de clasificar la información o los relativos a la continuidad del negocio; cumplimiento de la legislación vigente, especialmente las leyes relacionadas con las TIC.
2	Implantación y dimensionamiento justificado de los controles; análisis de los riesgos y políticas organizativas de las que se derivan procedimientos necesarios; formación en la seguridad, en el uso de los procedimientos o de técnicas, implantación de controles, etc.
3	Se completan los procedimientos que faltan, se comprueban los controles y se toman medidas correctoras; se añaden prácticas (opcionales) relativas al desarrollo de software seguro.

Tabla 2. Niveles de seguridad de la información.

2. Datos estadísticos de las evaluaciones ITMark

El estudio presentado en este apartado está basado en los datos de 47 evaluaciones realizadas en 44 organizaciones de 7 países (Figura 1). Tres de las organizaciones han sido evaluadas una segunda vez debido a la caducidad de la certificación anterior.

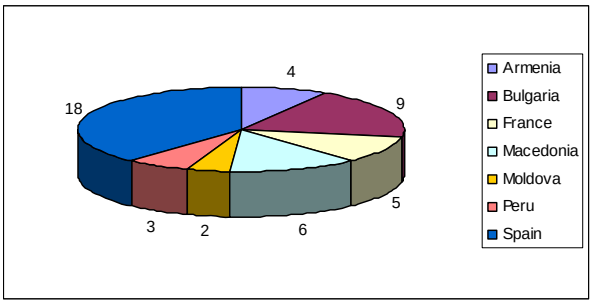


Figura 1. Número de organizaciones evaluadas por países.

El alcance de las evaluaciones incluía los procesos de negocio, de seguridad de la información y de las áreas de proceso de CMMI ML2.

En el 89% de las evaluaciones se ha alcanzado ITMark. Una empresa ha alcanzado ITMark Premium.

2.1. Categorías de negocio

Una visión general (Figura 2) de los resultados de las evaluaciones por categorías de negocio muestra que la dificultad principal está en la categoría de Estrategia y comité.

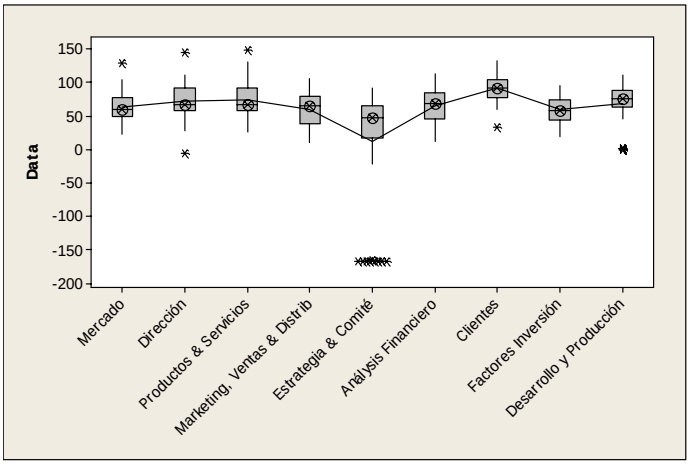


Figura 2. Resultados de las evaluaciones de las categorías de negocio.

Se ha hecho un análisis de los datos por categoría y de las correlaciones entre ellas. En cuanto a los datos por categoría, destacan dos aspectos (Figura 3):

- El 20% de las organizaciones evaluadas (9 de 47) no tiene un plan de negocio, por lo que ha obtenido resultado -166,7.
- El 13% de las organizaciones (6 de 47) no tiene establecido procesos de desarrollo y producción.

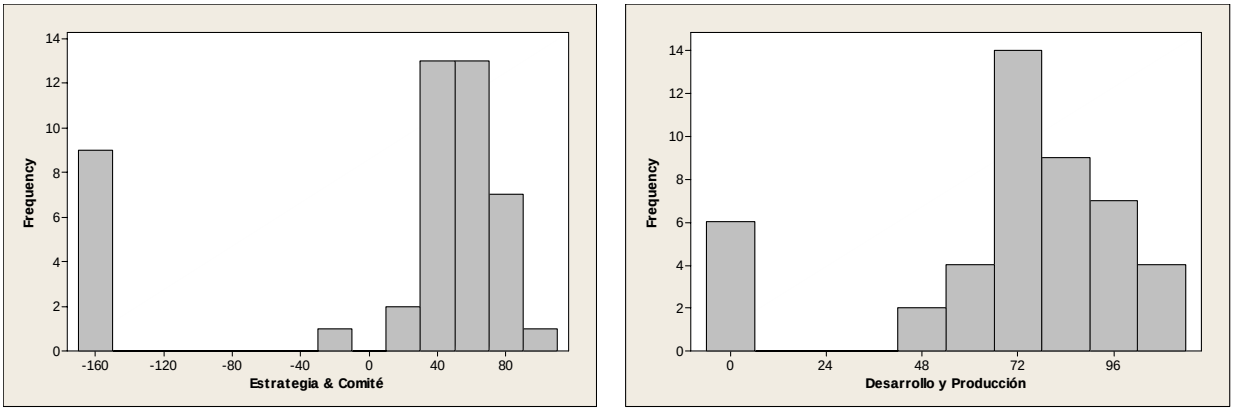


Figura 3. Histogramas de las categorías Estrategia y comité; Desarrollo y producción.

¿Es esto importante? ¿Qué consecuencias tiene carecer de plan de negocio y/o de procesos de desarrollo y producción?

Las graficas en la Figura 4 demuestran claramente que, cuanto mejor definidos están la estrategia y el plan de negocio de la empresa, mejores resultados se alcanzan respecto a la dirección y a los procesos de desarrollo y producción. Analizando en detalle el grupo de

empresas que no disponían inicialmente de un plan de negocio, hemos encontrado estas dificultades:

- La estrategia de la dirección no está definida, es confusa o no se conoce.
- Los productos no están alineados con las necesidades del mercado.
- Se desconoce el posicionamiento de la empresa respecto a sus competidores.
- Se realizan labores comerciales, pero sin una estrategia y un plan definidos.
- No se obtienen beneficios del desarrollo de los productos. Los proyectos se ejecutan sin formalismo.

Por lo tanto, dos de los aspectos fundamentales de la mejora en una empresa deben ser: la definición de su estrategia y plan de negocio, y el establecimiento de procesos de desarrollo y producción.

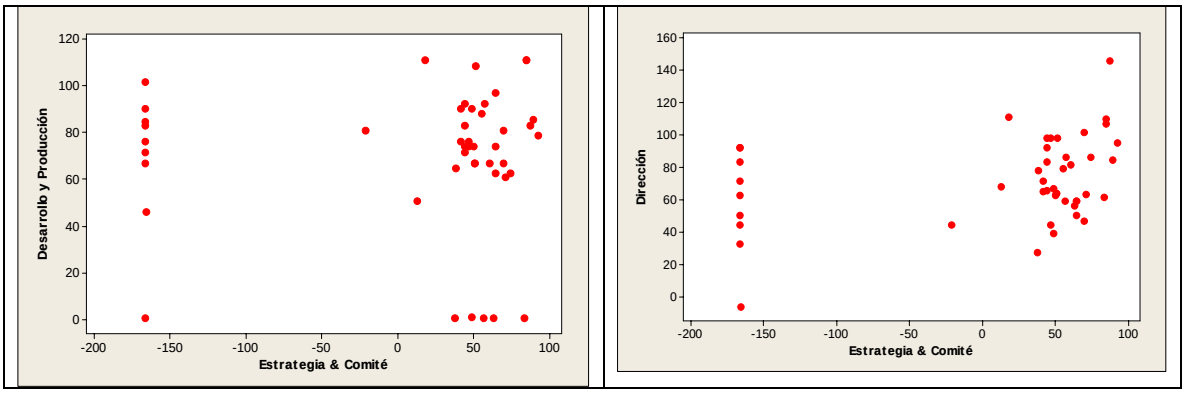


Figura 4. Dependencias entre estrategia y comité, y otras categorías.

Tal y como vemos en la Figura 5, la utilización de procesos bien definidos contribuye significativamente al desarrollo de productos y servicios de buena calidad. También es posible desarrollar buenos productos sin tener procesos definidos (ver los puntos a nivel 0 de desarrollo y producción), pero este es el caso típico de organizaciones que dependen bastante de las personas, de su experiencia, calificación, motivación y esfuerzo para alcanzar los objetivos de los proyectos de desarrollo. También se pueden ver casos de procesos bien definidos, pero poco eficaces para el desarrollo (ver los puntos a nivel 20-30 de productos y servicios). Esta situación se da en organizaciones donde las personas tienden a definir procesos más bien teóricos, pero poco aplicables en proyectos reales.

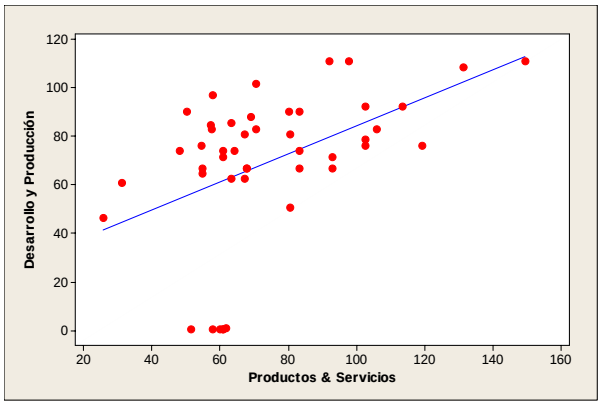


Figura 5. Dependencias entre las categorías productos y servicios, y desarrollo y producción.

2.2. Áreas de proceso del modelo CMMI®

Los procesos de desarrollo y producción de las organizaciones están evaluados con el modelo CMMI. Los resultados generales aparecen en la Figura 6.

La línea que corre a través del gráfico conecta las medias de cada área de proceso. Observamos que las áreas de proceso que sufren mayores variaciones son PPQA y MA.

En cuanto a PPQA, como vemos en la Figura 6 (b), el 23% de las organizaciones (11 de 47) no tiene este proceso desarrollado, ya que lo consideran un gasto no justificado. No obstante, el propósito de esta área de proceso es proporcionar una visión sobre el estado de los procesos. Además, como se ha visto en la Figura 5, la calidad de los productos desarrollados depende significativamente de los procesos que se aplican.

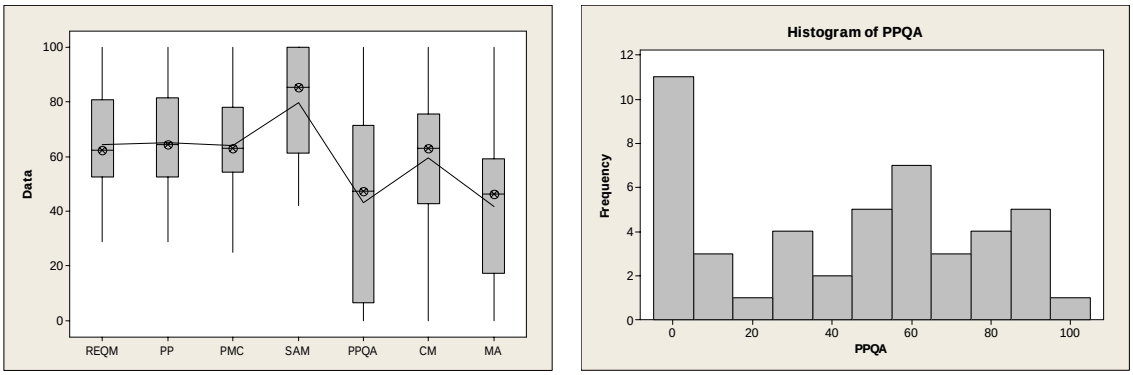


Figura 6. (a) Resultados de las evaluaciones CMMI. (b) Histograma de PPQA.

La razón principal para la variación en MA es que esta área de proceso requiere el establecimiento de un marco de medición que vincule los objetivos de negocio con las métricas que se utilizan en los proyectos. Además es importante que los datos se recojan y analicen correctamente. La Figura 7 demuestra claramente que la implementación del proceso MA contribuye a la realización de las actividades en la categoría estrategia y comité cuando el marco de medición y análisis definido en la organización sea común y tenga en consideración ambas áreas.

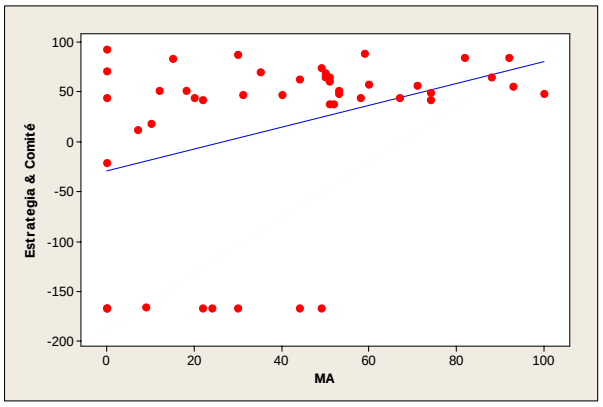


Figura 7. Dependencias entre MA y la categoría estrategia y comité.

2.3. Seguridad de la información

El 91% de las organizaciones han alcanzado el nivel 1 y el 4%, el nivel 2 de seguridad de la información. Se ha de destacar la falta de compromiso de la dirección en la "definición" de la seguridad. Es necesario integrar la gestión de la seguridad de la información en el sistema de calidad, de tal modo que la necesidad de la seguridad emane desde la dirección en forma de políticas que deriven en estándares, procedimientos y controles de seguridad.

3. Caso de estudio: ITMark Extremadura

Dos características definidoras de la región de Extremadura son la apuesta firme por el despliegue del software libre y la introducción de una cultura de calidad sólida en las pymes de la región. A través de la segunda, la Junta de Extremadura pretende aumentar la competitividad, la productividad y la rentabilidad de las pymes extremeñas. A finales del año 2006 la Junta lanzó el Proyecto de Mejora Competitiva del Sector TIC Extremeño.

A continuación se presentan datos reales que muestran la situación inicial en la que se encontraban las ocho pymes involucradas, y su evolución a lo largo del proyecto de mejora, hasta llegar, seis meses después, a una situación final favorable para todas.

3.1. Situación inicial

3.1.1 Procesos de negocio

Los resultados del diagnóstico inicial, basado en el método 10 squared, muestran (Figura 8) una carencia de atención significativa en áreas tales como mercado; dirección; *marketing*, ventas y distribución; estrategia y comité; análisis financiero; y desarrollo y producción. Esta es una situación usual en las pymes que están en la etapa de puesta en marcha, formadas mayoritariamente por emprendedores con elevado perfil técnico, pero con poca experiencia y formación en gestión y dirección de organizaciones. A continuación se detalla, por cada categoría, la casuística identificada:

- Mercado. Las organizaciones evaluadas no disponen, en el momento de la evaluación inicial, de la información necesaria para realizar un análisis de mercado, a pesar de que la información existe y está públicamente disponible. Hasta este momento, la gerencia toma decisiones en cuanto a la estrategia de mercado que se va a seguir, pero a partir de percepciones.
- Dirección. Es fundamental disponer de un equipo de dirección con la experiencia y las capacidades adecuadas para definir e implementar una estrategia empresarial. En general, no se dispone de una visión y misión clara de la organización, y en los casos en los que existe, esta no se comunica a toda la organización. Hay una fuerte dependencia de personas clave dentro de la organización.
- Marketing, ventas y distribución. Existe falta de dedicación a labores comerciales. Normalmente, la responsabilidad de estas actividades recae en la gerencia, en personas con un perfil muy técnico que, incluso, pueden estar involucradas en tareas técnicas y de gestión del día a día. No se dispone planes comerciales que detallen aspectos relevantes como las líneas de negocio, las previsiones anuales y futuras de ventas, los canales de distribución, etc.
- Estrategia y comité. Se identifica una falta de plan de negocio, documento fundamental, no solo para la creación o el lanzamiento de una organización, sino también para su gestión anual.
- Análisis financieros. No se dispone de sistemas de medición de indicadores y, en general, existen políticas de costes, aunque son informales.

- Desarrollo y producción. Se da una falta de definición de los procesos de la organización (gestión, desarrollo, etc.). También existen ciertos riesgos de producción ligados a la falta de capacidad de aumentar sustancialmente la producción en un plazo corto y con coste menor.

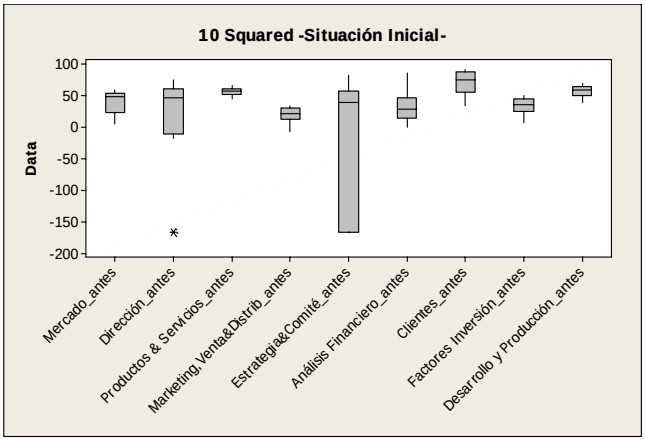


Figura 8. Situación inicial de procesos de negocio.

3.1.2. Procesos CMMI

Tras el diagnóstico inicial para cada una de las áreas de proceso, el resultado ha sido el que aparece en la Figura 9, que representa una situación propia de organizaciones no orientadas por procesos, con una fuerte dependencia de las personas de la organización y en las que se alcanzan buenos resultados, pero con un esfuerzo heroico. Se evidencia una carencia importante de atención en las áreas de PMC, MA (seguimiento del proyecto y de indicadores estratégicos alineados con los objetivos de la organización), PPQA (aseguramiento de la gestión por procesos) y CM (control de versiones).

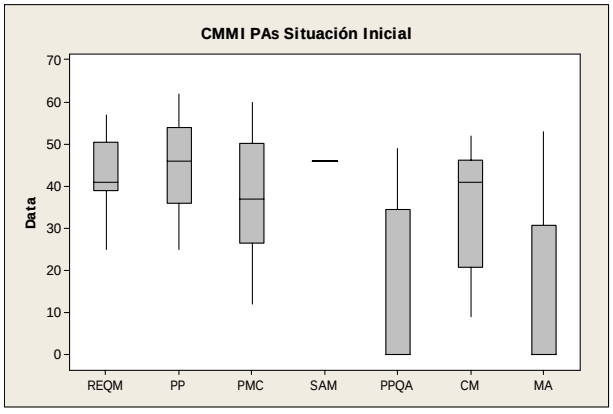
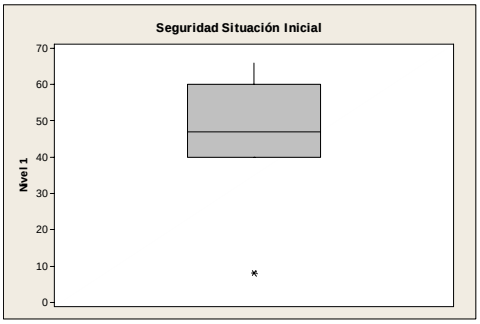


Figura 9. Situación inicial de CMMI.

- PMC. Se dispone de una percepción subjetiva sobre el estado de los proyectos, debido a que en muchas ocasiones no se realiza un seguimiento periódico de diversos parámetros como el esfuerzo, el porcentaje de avance de las tareas, etc. Además, no se dispone de una herramienta de imputación de horas a tareas planificadas en el calendario del proyecto. Estas organizaciones no están en disposición de anticiparse a los problemas, ya que no realizan una identificación y seguimiento formal de los riesgos del proyecto.
- MA. Estas organizaciones no disponen de un sistema de indicadores alineados con los objetivos estratégicos de la organización que sirva a la gerencia como herramienta para la toma de decisiones.
- PPQA. En este caso, encontramos dos casuísticas diferentes:
    1. Organizaciones que disponen de un SGC (Sistema de Gestión de Calidad). Existe la necesidad de reforzar la dinámica de las auditorías de calidad y objetivar su ejecución.
    2. Organizaciones que no disponen de un SGC implantado. No se realiza ninguna actividad de aseguramiento de la calidad y ni siquiera se percibe el valor de realizarla. Se requiere una actividad previa de concienciación sobre el porqué del aseguramiento de la calidad.
- CM. En general, se controlan los cambios en el código a través de herramientas de gestión de configuración existentes en el mercado como CVS, Subversion, etc. La documentación interna de la organización no se suele controlar; cuando surgen cambios, se actualiza una y otra vez la última versión.

3.1.3. Seguridad de la información



La seguridad, tal y como muestra la

Figura 10, es un área inmadura en las organizaciones, las cuales se excusan argumentando que no necesitan seguridad porque, por su tipo de negocio o por su presencia en el mercado, no van a ser objetivos de los delincuentes informáticos.

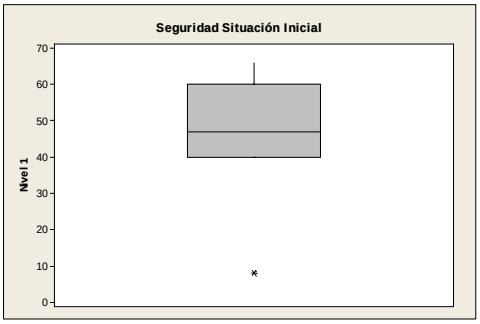


Figura 10. Situación inicial de seguridad.

- Compromiso de la dirección en la "definición" de la seguridad, según lo comentado en el apartado 2.3 del presente artículo.
- Ausencia de un listado de activos y propietarios/responsables de ellos mismos: las organizaciones no tienen identificados sus activos.
- Usuarios privilegiados: existen usuarios, se han definido controles de acceso, pero la mayoría de los usuarios trabaja con usuarios privilegiados, permisos de administrador. En alguna situación pueden ser necesarios permisos de usuario avanzado, pero trabajar continuamente con permisos de administrador significa introducir un riesgo grande.
- Clasificación de la información: ¿debería tratarse toda la información de la organización de igual forma? ¿Por qué se ha de invertir tiempo y dinero en proteger información que es pública? Lo primero que debe conocer una organización son los diferentes tipos de información que existen en ella. Una vez clasificados, se podrá decidir qué tratamiento se va a dar a cada uno de ellos.
- Cumplimiento de la legislación vigente. Las organizaciones conocen parcialmente las leyes que les afectan a nivel administrativo-financiero; sin embargo, otras relacionadas con las tecnologías de la información se desconocen o en los pocos casos en los que se conocen, no se suelen cumplir.

3.2. Situación final

Seis meses después del diagnóstico inicial, las organizaciones apoyadas con actividades de formación y soporte en la definición e implantación de las mejoras, logran una progresión significativa en la puntuación de cada una de las áreas (Figura 11).

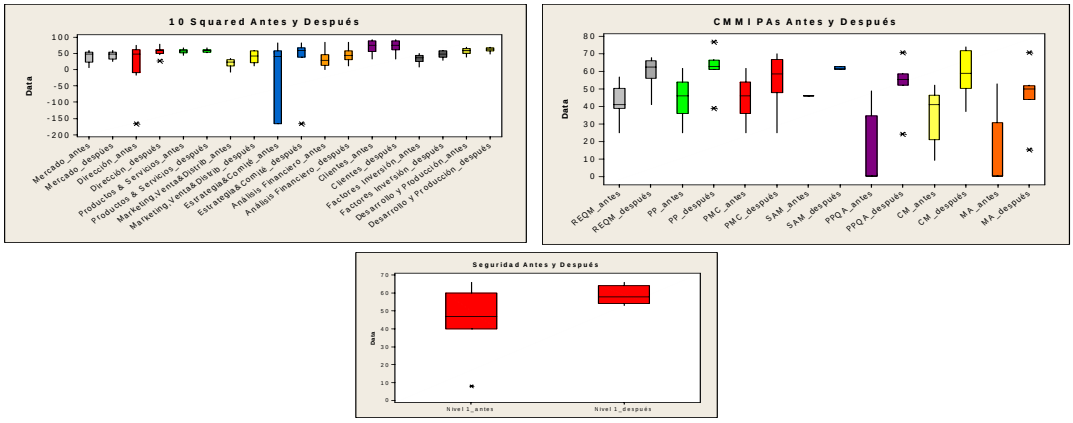


Figura 11. Situación inicial frente a situación final.

La dedicación por parte de las organizaciones en el proyecto de mejora, por término medio, es la siguiente:

- Gerencia: una persona con dedicación de tres días al mes.
- Equipo de proyecto: dos personas con dedicación de cuatro días al mes.
- Administración y sistemas: una persona con dedicación de dos días al mes.

4. Conclusiones

Hoy en día, para obtener mejores resultados de negocio, hay que mejorar, no solo los procesos técnicos de la organización, sino los de negocio y de la seguridad de la información. Las actividades de mejora en estas tres direcciones deben sincronizarse y mantenerse alineadas con los objetivos de negocio de la organización.

El análisis estadístico de los datos recogidos en 47 evaluaciones con ITMark, así como el caso de estudio ITMark Extremadura, demuestra que los puntos básicos en una mejora de procesos integrados y basados en diferentes modelos de calidad son:

- Desarrollo de una estrategia clara y un buen plan de negocio de la organización.
- Establecimiento de procesos contudentes de desarrollo y producción.

- Desarrollo de un marco de medición y análisis que dé soporte, tanto a la dirección de la organización, como a la gestión de los proyectos de desarrollo.
- Realización de actividades de aseguramiento de la calidad para conseguir que los procesos actuales sean útiles y efectivos.
- Compromiso de la dirección en la toma de decisiones, tanto técnicas como de gestión.

Referencias

[1] Chrissis, M. B., Konrad, M. y Shrum, S., *CMMI®: Guidelines for Process Integration and Product Improvement*, Software Engineering Institute. Carnegie Mellon University, 2006.

[2] ISO, *ISO/IEC 17799:2005 (27002). Information Technology. Security Techniques. Code of Practice for Information Security Management*, ISO, 2005.

[3] ISO, *ISO/IEC 27001:2005. Information Technology. Security Techniques. Information Security Management Systems. Requirements*, ISO, 2005.

Revisiones de código en el contexto del aseguramiento de calidad. Un caso práctico

M. J. Escalona, M. Pérez-Pérez, O. González-Barroso,
J. Ponce, J. M. Correa, A. I. Merino
Universidad de Sevilla
escalona@lsi.us.es

Abstract

The verification and validation of source code is one of the most critic tasks in quality assurance. Nowadays, development teams and technical offices are demanding valid tools and quality environments to perform verification and validation tasks. Nowadays, there are several powerful tools which allow to automatism these tasks. This paper introduces a practical case using PMD tool in real software projects.

Key words: quality of software, verification and validation of source code.

Resumen

La verificación y validación del código es una de las actividades más críticas en los desarrollos de software que sirven para verificar la calidad de los productos que se generan. En la actualidad, los equipos de desarrollo y las oficinas de calidad demandan herramientas y entornos de referencia adecuados para esta verificación y validación. Hoy en día existen entornos de herramientas potentes y que permiten automatizar este tipo de tareas. En este trabajo se presenta una muy concreta, PMD, y se ofrece una visión de cómo se ha adaptado a diferentes proyectos reales.

Palabras clave: calidad del software, verificación y validación de código.

1. Introducción

Cada día, la complejidad de los proyectos de software aumenta y se incrementa la necesidad de controles e inspectores que permitan medir la calidad de los trabajos que se van realizando sin llegar hasta el final del proceso [1].

Detectar los errores en las etapas más tempranas posibles garantiza un menor coste de resolución; por ello, hoy en día las empresas están potenciando el uso de herramientas y entornos que permitan hacer una medida automática de estos indicadores.

El grupo de autores de este artículo orienta su línea de investigación hacia estos aspectos. Sin embargo, este trabajo se centra una propuesta para analizar la calidad de las entregas en la fase de construcción.

La herramienta PMD incluye una serie de reglas y permite validar el código Java de un proyecto. La herramienta, de software libre y que se integra fácilmente en entornos como eclipse, permite aplicar una serie de reglas para medir y validar la calidad del código Java. Estas características, además de la existencia de una comparativa publicada en la ISSRE 2004 [2], hicieron que se eligiera PMD frente a otras propuestas. Sin embargo, la configuración por defecto de PMD no siempre resulta operativa en todo tipo de proyectos.

En el epígrafe 2 de este trabajo se presenta la tecnología PMD; luego se continúa analizando cómo puede adaptarse a proyectos reales y se explica la experiencia que se ha obtenido en proyectos reales. Con este fin, en el apartado 3 se analiza la adaptación que se ha realizado y en el apartado 4 se presenta la visión práctica y la experiencia obtenida. Finalmente, se termina con un análisis de los trabajos relacionados y con las conclusiones y trabajos futuros.

2. La tecnología PMD

El objetivo principal de una herramienta como PMD es ofrecer un entorno automático que permite la obtención de métricas objetivas para medir la calidad de un código fuente. En este contexto, se entiende por métricas de código a un conjunto de medidas de software que proporcionan a los programadores una mejor visión del código que están desarrollando.

Normalmente, la calidad y la velocidad de desarrollo no van unidas y la perfección tiene, en la mayoría de los casos, un coste demasiado alto. De esta forma, se tiende a construir software de acuerdo con unos tiempos y unos costes que en muchos casos son tan irreales que hacen que la calidad de los productos obtenidos se vea mermada.

Herramientas como PMD tratan de definir entornos normalizados que ayuden a mejorar la integración y herencia de código ajeno, así como a facilitar el establecimiento de un entorno común de desarrollo, favoreciendo la comprensión del código y, sobre todo, la minimización del impacto de los errores de integración.

Con este objetivo, el primer problema que hay que solventar es cuantificar la calidad de un código fuente Java, lo cual puede llegar a ser algo bastante subjetivo. Por esta razón, es necesario buscar mecanismos que aseguren una homogeneidad de criterios y, por tanto, de los resultados de la evaluación, independientemente de si el código es evaluado

por el propio grupo de desarrollo, por un grupo externo de calidad o por cualquier otro miembro del proyecto.

Se hace necesario que la validación del código, o bien se realice de acuerdo con una lista de comprobación bien definida (prueba, error y valoración objetiva), o que se haga de un modo automático.

A tal fin nacen productos de software como PMD, que automatiza esta validación y, por tanto, nos permite garantizar esta homogeneidad.

PMD es totalmente integrable en entornos como NetBeans o Eclipse y, tal como se analiza en el apartado de trabajos relacionados, no excluye a otros. Entre sus características fundamentales se encuentra su capacidad de analizar código desde el punto de vista de la ejecución y su capacidad de detectar:

- Posibles errores: try/catch/finally/switch incompletos.
- Código que no se usa: variables locales, parámetros y métodos privados.
- Código no óptimo: mejor uso de String/StringBuffer.
- Expresiones redundantes: if innecesarios, for loops que podrían ser while loops, etc.
- Código duplicado: código copiado y pegado significa bugs copiados y pegados.

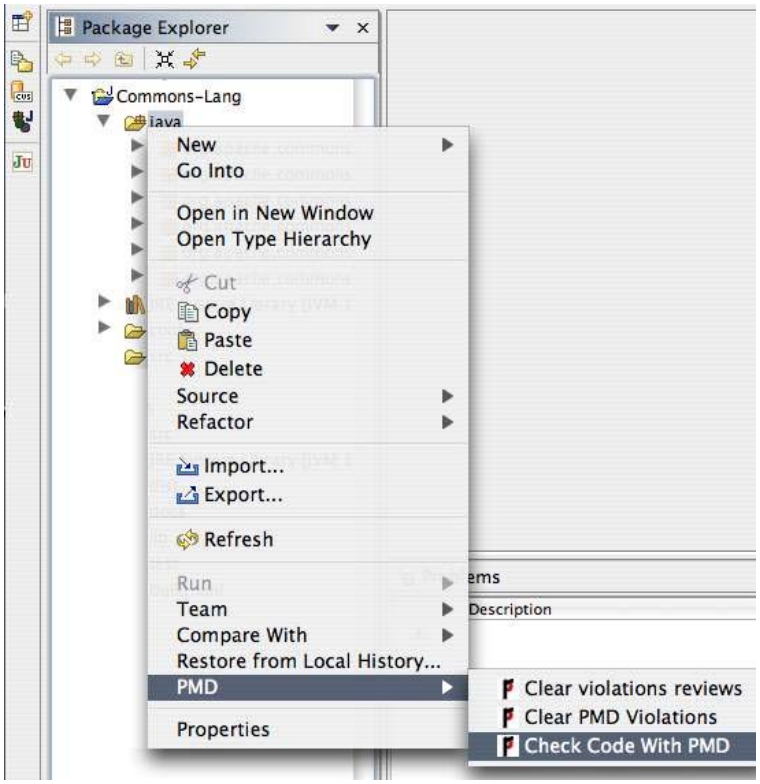


Figura 1. Interfaz de PMD.

Además, por ser un producto de software libre, está integrado con los principales IDEs[1]: JDeveloper, Eclipse, jEdit, JBuilder, BlueJ, CodeGuide, NetBeans / Sun Java Studio Enterprise / Creador, IntelliJ IDEA, Textpad, Maven, Ant, gel, JCreator y Emacs, de los cuales existen abundantes manuales para cada IDE en las web habituales de consultas de desarrolladores.

También cabe resaltar el soporte ofrecido en la web oficial (http://sourceforge.net/project/showfiles.php?group_id=56262), desde donde se puede descargar la última versión liberada, un resumen de todas las reglas (Current Rulesets), documentación específica, etc.

En la Figura 1 se muestra, a modo ilustrativo, la sencillez con la que PMD se integra dentro del propio eclipse.

3. La adaptación de PMD

La herramienta PMD presenta una serie de reglas definidas. Sin embargo, para ofrecer un mayor rendimiento en los proyectos es más adecuado redefinir nuevas reglas y hacer un estudio de aquellas que resultan más convenientes para el entorno en el que se trabaje.

Uno de los objetivos que nos marcamos como grupo de investigación es ofrecer entornos para evaluar el código fuente para la fase de construcción e implantación de proyectos. Decidimos hacer uso de PMD para detectar posibles errores y código ineficiente. El uso de PMD requiere una serie de adaptaciones al trabajo.

El proceso para definir el entorno más adecuado a los proyectos en los que colaboramos siguió una secuencia de pasos. En un primer lugar, había que seleccionar la plataforma, PMD, que puede ejecutarse sobre la línea de comandos, pero también ofrece la posibilidad de integrarse en plataformas de desarrollo. En nuestro caso, la elección fue Eclipse, puesto que era el entorno de desarrollo que más se utilizaba en aquellos entornos en los que trabajábamos. De esta forma, se propuso el plugin que facilita PMD mediante el cual se integra dentro del entorno Eclipse (http://pmd.sourceforge.net/integrations.html#eclipse).

[1] Para todas las herramientas que se mencionan en este artículo, consúltese su URL en el Apéndice A.

El siguiente paso era definir las reglas que, de manera efectiva, había que utilizar para la implantación. PMD ofrece un conjunto de reglas predefinidas (Rulesets) en el que se incluye un gran número de reglas cuya criticidad es evaluada desde el 1, muy críticas, hasta el 5, leves.

Para decidir y adecuar estas reglas, se evaluó sobre un proyecto real el alcance de los errores detectados con los Rulesets originales de PMD. Tras un estudio de esos errores, se hizo un estudio para adaptar esos Rulesets a nuestras necesidades, adoptando nuevas reglas de verificación de código y eliminando aquellas que, de manera empírica, parecía que detectaban errores no relevantes al entorno de trabajo.

Las reglas en PMD se pueden definir con código Java o con XPath (http://www.w3.org/TR/xpath). Las reglas incluidas en nuestra adaptación han sido definidas de ambas maneras, puesto que en algunos casos sencillos era más fácil definir la regla con Java, mientras que para reglas con una alta complejidad, la mejor opción era utilizar XPath.

Para añadir nuestras propias reglas, hemos usado la herramienta de diseño de reglas que viene incluida con PMD. El proceso para crear una regla se resume en escribir un trozo de código que incumple dicha regla, generar con la herramienta el árbol de sintaxis abstracta y escribir la expresión XPath que encuentre solo los nodos del árbol que incumplen la regla. La misma herramienta de diseño de reglas de PMD genera el código XML de la regla para poder incluirla dentro del Ruleset.

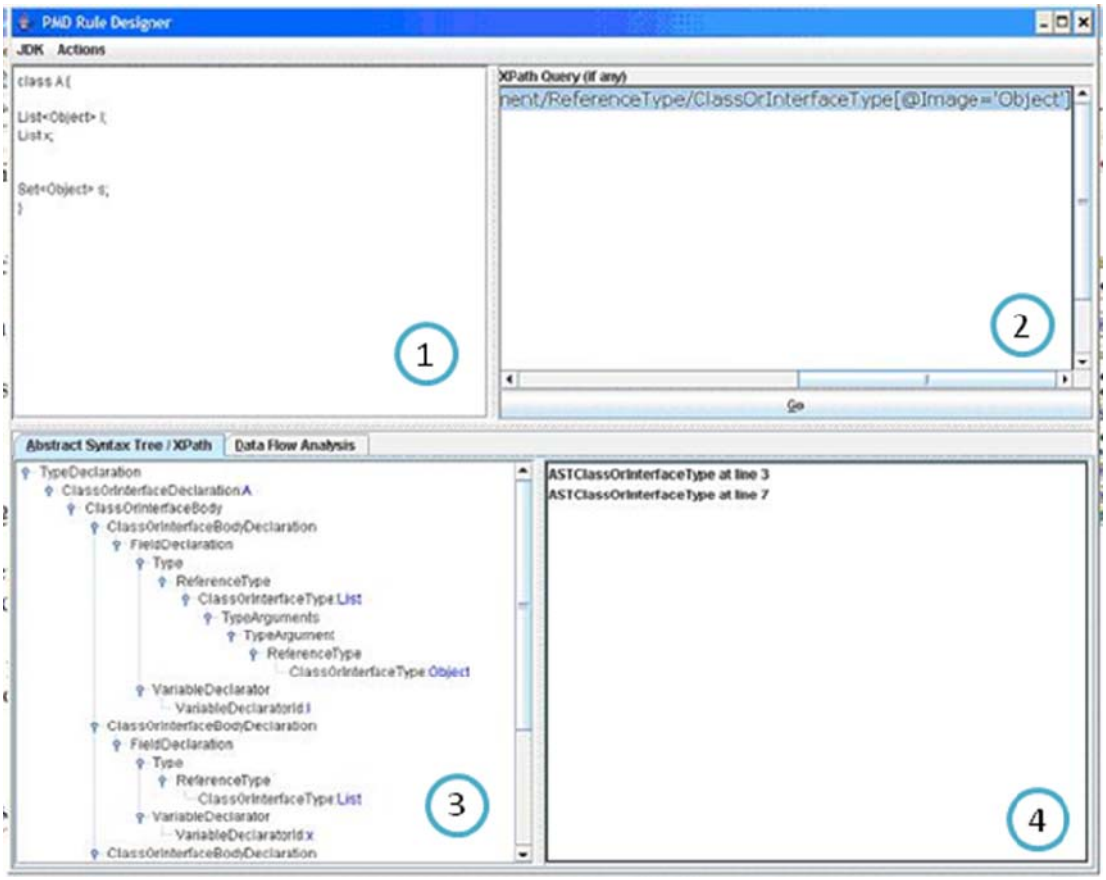


Figura 2. Diseño de una nueva regla para PMD.

En la Figura 2 se muestra una captura de la herramienta de diseño de reglas con una nueva regla que verificará que no se utilicen genéricos de tipo Object (por ejemplo, List<Object>). En el cuadro 1 de la Figura 2 se ha escrito un fragmento de código Java que tiene dos incumplimientos de la regla (las líneas “List<Object> l;” y “Set<Object> s;”). En el cuadro 3 se puede ver el árbol de sintaxis abstracta generado a partir del código de ejemplo del cuadro 1. En el cuadro 2, se muestra parcialmente una expresión XPath que permite encontrar aquellos nodos del árbol que declaran genéricos de tipo Object. Esa expresión se muestra a continuación:

```
//ReferenceType/ClassOrInterfaceType/TypeArguments/TypeArgument/ReferenceTy
pe/ClassOrInterfaceType[@Image='Object']
```

Finalmente, en el cuadro 4 se puede comprobar cómo la expresión detecta aquellas líneas que declaran genéricos de tipo Object.

Tras estudiar los diferentes niveles de error definidos en el Ruleset original, decidimos que, para los proyectos que evaluamos, los errores de nivel 1 y 2 eran inadmisibles para la aceptación del código. Los errores de nivel 3 serían revisados manualmente, ya que algunos podrían ser aceptables o no, y los restantes errores de nivel 4 y 5 serían aceptados.

Otro de los pasos que hay que seguir en la adaptación es la parametrización de la herramienta CPD (Copy&Paste Detector) (http://pmd.sourceforge.net/cpd.html). Esta herramienta busca en el código trozos duplicados. La parametrización de la herramienta se ha modificado a 50 caracteres (en su versión por defecto está en 25 caracteres), ya que es más que suficiente para detectar bloques de 5 o 6 líneas si estas tienen muchos caracteres, o unas 10 líneas si tienen pocos.

Y, por último, había que adecuar la salida de la evaluación realizada por PMD. A la hora de presentar un informe de la evaluación de la calidad del código, en un principio usamos la opción que ofrece el plugin de Eclipse de exportar el reporte de incidencias a CSV y posteriormente lo importamos desde una base de datos Access. En esa base de datos procedíamos a ejecutar consultas y a generar informes con los diferentes errores, según paquete de código y según nivel de criticidad.

Posteriormente, tras entrar a fondo en la propia herramienta PMD mediante línea de comando, decidimos hacer uso del paquete Renders (http://net.sourceforge.pmd.renderers), haciendo una ampliación de esta funcionalidad mediante la creación de una clase que genera un documento RTF con la librería iText (http://www.lowagie.com/iText/), consistente en una tabla con todas las incidencias encontradas por PMD, que muestra el archivo en el que está el error, la línea, el mensaje y el nivel de criticidad del error.

4. Aceptación de PMD

Tras adaptar PMD a nuestra forma de trabajo para conseguir un grado de optimización de código Java basado en el estándar más actual, había que hacerlo llegar a las empresas. De esta forma, colaborando con diferentes entidades, hemos implantando el uso de PMD en la evaluación de 22 proyectos de software, de diferente tamaño y complejidad, en la

Consejería de Cultura de la Junta de Andalucía y en un proyecto de gran envergadura de la Empresa Municipal de Aguas de Sevilla, Emasesa.

Ante el desconocimiento de la herramienta, en cierto modo encontramos cierta resistencia ante los resultados obtenidos por PMD, puesto que suponía un atraso en la entrega de sus proyectos, debido en gran parte al tiempo que empleaban los equipos de desarrollo de estas entidades para corregir los errores detectados por PMD.

Se impartieron cursos de formación en los que se trataban temas como la compatibilidad de PMD para ser instalado en muchas de las plataformas de trabajo (en nuestro caso, Eclipse), así como su uso a la vez que se desarrollaba el código, lo cual evitaba, en gran medida, muchos de los errores con los que llegaba el proyecto.

La finalidad de estos cursos era convencer a los diferentes jefes de proyectos de la implantación de esta herramienta a sus equipos de desarrollo para que, al tiempo que se iba desarrollando la herramienta, esta se pudiera ir ejecutando y se fueran corrigiendo los errores detectados de una forma limpia. De este modo, aquel código que provenía de entornos de desarrollo que venía aplicando la herramienta, era entregado totalmente limpio de errores inaceptables (errores y errores altos o graves), o con un número bajo de errores de otros niveles que se consideran admisibles desde el punto de vista de la calidad.

La transición desde el uso de esta herramienta ha sido buena, puesto que al inicio la gran mayoría de los proyectos que se entregaban tenían una calidad muy baja, a causa de numerosos errores graves (que no eran permitidos dentro de nuestro baremo) y de la no utilización de la herramienta.

Actualmente la forma de trabajo es buena, puesto que los diferentes proyectos que aún carecen de esta herramienta como un valor añadido a su proyecto desde el punto de la vista de la calidad del código, se van pasando por estas verificaciones cada poco tiempo. De este modo se van solucionando los problemas que van surgiendo, en lugar de realizar una única entrega que no haya pasado progresivamente bajo los estudios de calidad realizados por la herramienta PMD. Por el contrario, aquellas empresas que cuentan con esta herramienta como base para aplicar diferentes criterios de calidad a su código, llegan con mínimos errores y se devuelven muy pocos proyectos con incidencias serias que, como ya comentamos antes, en los niveles de gravedad de PMD serían errores graves y muy graves.

5. Trabajos relacionados

Como ya se ha mencionado con anterioridad, existen distintas herramientas que permiten estudiar métricas de código Java y revisar conjuntos de reglas. Al consultar distintos buscadores de Internet, tanto genéricos (como Google) como específicos en el ámbito de la programación (como Sourceforge), se han encontrado 18 herramientas, 9 de ellas de libre descarga y uso, y 9 comerciales.

Herramientas gratuitas	Enlaces
Checkstyle	http://checkstyle.sourceforge.net/
DoctorJ	http://www.incava.org/projects/java/doctorj/
ESC/Java	http://kind.ucd.ie/products/opensource/ESCJava2/
FindBugs	http://findbugs.sourceforge.net/
Hammurapi	http://www.hammurapi.biz/
JCSC	http://jcsc.sourceforge.net/
Jikes	http://jikes.sourceforge.net/
JLint	http://www.artho.com/jlint/
JWiz	http://csdl.ics.hawaii.edu/Research/JWiz/JWiz.html

Tabla 1. Herramientas gratuitas.

Herramientas de pago	Enlaces
AppPerfect	http://www.appperfect.com/
Assent	http://www.pune.tcs.co.in/Assent_registration.htm
Aubjex	http://mindprod.com/jgloss/aubjex.html
AzoJavaChecker	http://www.andiz.de/azosystems/en/index.html
CodePro Studio	http://www.instantiations.com/codepro/index.html
Flaw Detector	http://www.excelsior-usa.com/fd.html
JStyle	http://www.mmsindia.com/jstyle.html
JTest	http://www.parasoft.com/jsp/products/home.jsp?product=Jtest
Lint4J	http://www.jutils.com/

Tabla 2. Herramientas de pago.

Dado que aquí es imposible describirlas todas en detalle, a continuación se citan algunas de las herramientas gratuitas más relevantes y se comparan con la herramienta PMD.

5.1. Checkstyle

Como ya se ha indicado, esta herramienta verifica el código fuente para asegurar que este sigue un estándar de codificación. Sin embargo, el número de reglas que incorpora es menor que PMD, muchas de las reglas ya están incluidas en PMD y las que no lo están son

muy sencillas de añadir, como se ha visto en secciones anteriores. Sin embargo, la herramienta Checkstyle es más difícil de extender, ya que solo ofrece la opción de extenderse escribiendo código Java y aplicando la implementación del patrón visitante (VP).

5.2. Hammurapi

La documentación de esta herramienta es inferior a la documentación de la herramienta PMD. Además, también es necesario escribir nuevo código Java para extender el conjunto de reglas.

5.3. JLint

A diferencia de PMD y de las herramientas anteriores, la herramienta JLint está desarrollada en C++. Al igual que las herramientas anteriores, es más difícil de extender e incluye menos reglas que PMD. Experimentos publicados con anterioridad también muestran que esta herramienta tiene un porcentaje de falsos errores más alto que PMD. Sin embargo, JLint sí es capaz de verificar el tamaño de las tablas y detectar tamaños de tablas menores de cero, lo cual PMD no es capaz de hacer.

5.4. JCSC

Esta herramienta tiene más opciones de análisis que PMD, como el control de la documentación, la nomenclatura y el orden de la definición de las clases. JSCS tiene menos plugins que PMD, si bien tiene un plugin completo en ejecución de código que generará un documento con los errores dado por el código.

5.5. Assent

Assent es una herramienta comercial alternativa a PMD. No obstante, existe un plugin de Assent para Eclipse que nos permitiría usar las reglas de PMD complementadas con Assent y viceversa. Tiene una amplia gama de control en documentación de clases, métodos y variables.

6. Conclusiones

Este artículo se centra en una fase muy concreta del ciclo de vida, la construcción. En él se presenta el uso de la herramienta PMD como solución para evaluar de una manera objetiva la calidad de los productos de software entregados.

El artículo ha comenzado haciendo una breve introducción de PMD, enfocada a sus objetivos y características principales. Tras eso, se ha presentado el proceso seguido por el grupo de autores para realizar su adaptación y adecuación al entorno real de los proyectos con los que comúnmente trabajan.

Después de la presentación de la adaptación, se explicado cómo se ha implantado en los proyectos y los costes en formación que esto ha supuesto. Finalmente, se han presentado los trabajos relacionados.

Cabe decir que, tras las experiencia real, y una vez formado el equipo de desarrollo, PMD no es un herramienta que encarezca los proyectos. Si los equipos de programadores conocen las reglas de antemano, poco a poco van adquiriendo aptitudes que los orientan a escribir el código, incluyendo esos aspectos de calidad.

Por otro lado, nuestro trabajo de adaptación no ha terminado aún. Si bien el conjunto de reglas que se propone para los proyectos está ya muy depurado, con cada nueva aplicación empírica encontramos una nueva fuente de inspiración para la revisión. En la actualidad se aplica a 23 proyectos reales y a proyectos fin de carrera que desarrollamos en el seno del grupo de investigación. No obstante, está previsto que este número crezca gracias a las colaboraciones con empresas que mantenemos.

Por último, hemos de indicar que la aplicación de PMD es solo una parte del trabajo de calidad que proponemos desde el grupo. En él tenemos definida toda una metodología de desarrollo en la que existen inspectores y medidores de calidad en todas las fases y que también aplicamos sobre estos proyectos. Actualmente estamos en plena fase de certificación, bajo la norma ISO 9001, de todo el sistema de aseguramiento de la calidad que tenemos definido.

Agradecimientos

La elaboración de este artículo ha contado con el apoyo del subproyecto QSimTest del proyecto Cycit (TIN2007-67843-C06_03) y de la red de investigación RePRIS (TIN2007-30391-E) del Ministerio de Educación y Ciencia.

Referencias

[1] Armesto, A., Loniewski, G., Carlos, A. y Llorens, V., “Incorporando el análisis estático de código en un proceso de integración continua”, en: Vos, T. J. (ed.), *Proceedings de las*

IV Jornadas de Testeo de Software 2007. Valencia (España), 3-4 de mayo de 2007, pp. 65-78, 2007.

[2] Rutar, N., Almazan, C. B. y Foster, J.S., "Comparison of Bug Finding Tools for Java", en: *Proceedings of the 15th International Symposium on Software Reliability Engineering. Saint-Malo (France), 2-5 de noviembre de 2004*, pp. 245- 256, 2004.

Apéndice A

JDeveloper: http://www.oracle.com/technology/products/jdev

Eclipse: http://www.eclipse.org/

jEdit: http://www.jedit.org/

JBuilder: http://www.codegear.com/products/jbuilder

BlueJ: http://www.bluej.org/

CodeGuide: http://www.omnicore.com/

NetBeans: http://www.netbeans.org/

IDEA: http://www.jetbrains.com/idea/

Textpad: http://www.textpad.com

Maven: http://maven.apache.org/

Ant: http://ant.apache.org/

Gel: http://www.gexperts.com/gel

JCreator: http://www.jcreator.com/

Emacs: http://www.gnu.org/software/emacs/

Diagnóstico de la situación de la calidad del software en la industria española

Elena Argüelles, Antonio Sepúlveda
Laboratorio Nacional de Calidad del Software.
Instituto Nacional de Tecnologías de la Comunicación (INTECO)
{elena.arguelles, antonio.sepulveda}@inteco.es

Abstract

If we analyse the situation of software quality in the Spanish industry, we can see that there is great ignorance about the real benefits and a lack of advertising of the existing certificates and role models. Therefore, an intermediary agent is necessary to bring together initiatives in the field of quality and to advertise and add value to the quality certificates for clients and companies. In addition, a widespread lack of knowledge can be seen with regard to the existing methodological proposals. Furthermore, despite that the use of tools has become a necessity for companies, few organisations use them to cover the entire software project. Thus, it is necessary for the different actors involved (companies, professionals and public institutions) to carry out activities related to the dissemination and implementation of methodologies and tools applicable to the software lifecycle processes. Finally, we present an analysis of weaknesses, threats, strengths and opportunities which distinguish Spain as *nearshore* destination for software factories, and which role companies and public administrations can play in supporting a software factory model with a *nearshore* approach.

Key words: certification, methodologies, tools, software factories.

Resumen

Tras el análisis de la situación de la calidad del software en la industria española, se observa que existe un amplio desconocimiento acerca de los beneficios reales y una falta de publicidad sobre los modelos que se deben seguir y las certificaciones existentes. Se hace necesaria la presencia de un agente intermedio que aglutine las iniciativas en el ámbito de la calidad, dando publicidad y valor a las certificaciones ante clientes y empresas. Además, se evidencia una falta de conocimiento de las propuestas metodológicas existentes y que, a pesar de que su uso ha llegado a convertirse en una necesidad para las empresas, pocas organizaciones usan herramientas que cubran todo el proyecto de software. Es necesario realizar actividades de difusión e implantación de las metodologías y herramientas aplicables a los procesos del ciclo de vida del software por parte de los diferentes agentes implicados (empresas, profesionales y entidades públicas). Por último, se identifican las debilidades, amenazas, fortalezas y oportunidades que caracterizan a España como destino *nearshore* para factorías de software y se señala el papel que pueden desempeñar empresas y administraciones públicas para apoyar el modelo de factorías de software con un enfoque *nearshore*.

Palabras clave: diagnóstico, certificación, metodologías, herramientas, factorías de software.

1. Introducción

INTECO (Instituto de las Tecnologías de la Comunicación, S.A.) es una sociedad estatal adscrita al Ministerio de Industria, Turismo y Comercio a través de la Secretaría de Estado de Telecomunicaciones y para la Sociedad de la Información. Se ha concebido como instrumento de desarrollo de la sociedad de la información en España, para lo cual gestiona, asesora, promueve y difunde diferentes proyectos enmarcados en la estrategia del gobierno que se contiene en el plan Avanza.

INTECO, a través de su línea de calidad del software, promueve estrategias en materia de calidad en la industria española del software, así como el desarrollo y asentamiento de proyectos empresariales en España. Entre sus objetivos está el diagnóstico de la situación de partida en el ámbito de la calidad del software y de la certificación en la industria española de las tecnologías de la información y las comunicaciones (TIC), analizando el impacto que la calidad del software tiene en la competitividad de este sector. En este sentido, hasta el momento se han llevado a cabo tres estudios, de los cuales se presenta su diagnóstico y se ofrecen unas recomendaciones.

2. Estudio sobre la certificación de la calidad como medio para impulsar la industria de desarrollo del software en España

El propósito final del estudio es evaluar la situación actual de la certificación de calidad en el desarrollo de software, tanto en lo que respecta al mercado de certificaciones y modelos, como en lo referente a las implantaciones de estos.

En este trabajo se han realizado entrevistas a personas responsables de los procesos de calidad del software y/o personas con experiencia en este ámbito, pertenecientes a 28 entidades seleccionadas por considerarse un referente en relación con la gestión de la calidad del software. Pertenecen a las siguientes categorías: pymes, grandes empresas, asociaciones empresariales, organismos públicos, entidades certificadoras, entidades normalizadoras y acreditadoras, expertos de universidades españolas y expertos independientes no vinculados a ninguna de las categorías anteriores.

2.1. Diagnóstico

El sector de desarrollo de software en España se compone de pymes en un 99,8%, de las que más del 85% son microempresas de menos de 10 empleados. Según datos del Instituto

Nacional de Estadística (INE), en 2006 existían 32.023 empresas dedicadas a actividades informáticas, de las cuales 31.905 (el 99,63% del total) son pymes, lo que supone que estas empresas son el principal motor de la actividad informática en España. Estas pymes generan más del 60% del valor añadido en el sector y el 70% de los puestos de trabajo. Un elemento de éxito fundamental es la calidad, lo que hace de la adopción de una disciplina en calidad un hecho vital para conseguir ventaja competitiva en la industria del software en España.

Según los resultados de las entrevistas realizadas a pymes, existe un desconocimiento general en España acerca de las ventajas que puede aportar la certificación a una pyme. El problema fundamental de los modelos más conocidos y publicitados en el mercado es que están pensados para grandes empresas. Además, su implantación conlleva costes económicos muy elevados.

Prácticamente todos los entrevistados coinciden en señalar como solución más lógica la creación de un modelo reducido orientado expresamente a este tipo de entidades.

2.2. Recomendaciones

Respecto a la señalada necesidad real en España, se puede afirmar que, a priori, no existe un motivo de peso que impulse a una pyme hacia la implantación de un modelo de mejora; este aspecto es más evidente cuanto más pequeña es la empresa. Se hace necesaria la figura de un agente intermedio que sirva como ente que publicite y dé valor a las certificaciones de calidad, realizando campañas de información dirigidas tanto a las empresas como a sus clientes. Además, este agente intermedio debería aglutinar las iniciativas en el ámbito de la calidad y servir como puente entre la pyme y las entidades normalizadoras y organismos que respaldan la creación de estándares, normas y modelos. Este agente intermedio debería:

- Identificar los modelos existentes para los diversos tipos y tamaños de empresas, seleccionando aquellos que mejor se adapten a cada una.
- Fomentar la creación y el desarrollo, en el caso de que los modelos sean insuficientes para algún tipo o tamaño de empresa, de modelos adecuados a la idiosincrasia de cada una de ellas.
- Promover y divulgar iniciativas para difundir los modelos entre las empresas.

- Crear un directorio de contactos y un repositorio de información pública respecto a los agentes implicados en los esquemas de evaluación o certificación de cada modelo.
- Aglutinar y respaldar iniciativas de otras entidades, públicas o privadas, con el mismo fin.
- Gestionar, como punto único de información respaldado, reconocido y reconocible, la información referente a planes de futuro, subvenciones y/o créditos ofrecidos por entidades públicas o privadas para llevar a cabo implantaciones de estos modelos.
- Realizar las oportunas campañas de información para hacer ver a los compradores de software en España la necesidad de exigir productos con un sello de calidad.
- Divulgar entre las microempresas españolas los beneficios reales de la implantación de modelos para generar productos bajo un sello de calidad.

Los entrevistados en este estudio identifican a una entidad pública como el organismo que debería erigirse como agente impulsor de la certificación.

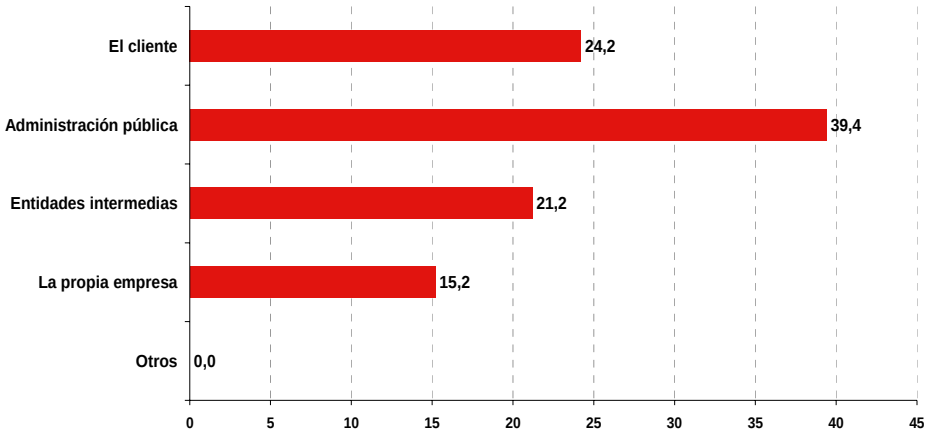


Figura 1. Entidades u organismos que deben impulsar la adopción de modelos de calidad por parte del empresariado español.

3. Estudio sobre metodologías y herramientas empleadas en los proyectos de software en España y su repercusión en la calidad de los productos y servicios finales

El estudio ofrece una visión del estado de salud de las empresas españolas a nivel de calidad del software, en el uso y conocimiento tanto de metodologías de trabajo, como de herramientas, desde la perspectiva de la oferta y la demanda.

Para realizar este trabajo se han llevado a cabo 2.925 encuestas en todo el territorio nacional, distribuidas entre los siguientes ámbitos:

	Ámbito sectorial	**Ámbito profesional**
Oferta	Actividades informáticas Investigación y desarrollo	Ingenieros superiores Profesionales de la informática de nivel superior, medio y técnico
Demanda	Industria Construcción Servicios Administración pública, defensa y seguridad social obligatoria Otras actividades profesionales	Responsables de compras

Tabla 1. Ámbito de estudio.

Por otro lado, se han realizado 75 encuestas a factorías de software españolas.

3.1. Diagnóstico

3.1.1. Conocimiento de las metodologías y nivel de uso

Del análisis de los resultados de la investigación se deduce un amplio desconocimiento de las propuestas metodológicas existentes. Así, dos de cada tres encuestados de la demanda (64,8%) y uno de cada tres de la oferta (37,6%) afirman no tener conocimiento alguno sobre los estándares, las normas oficiales ni las metodologías orientadas a la calidad.

Nº de trabajadores	**Sí**		**No**		**No sabe**		**No contesta**	
	Oferta	**Demanda**	**Oferta**	**Demanda**	**Oferta**	**Demanda**	**Oferta**	**Demanda**
Menos de 10	35,1	17,5	52,7	75,9	6,5	5,0	5,7	1,6
De 10 a 49	49,0	23,6	41,0	71,2	5,3	3,1	4,7	2,1
De 50 a 250	61,6	42,0	29,8	49,7	3,0	3,8	5,6	4,5
Más de 250	68,3	64,8	17,2	24,7	7,1	6,2	7,4	4,4

Tabla 2. Conocimiento de estándares, normas oficiales y metodologías según el tamaño de la organización, tanto en oferta como en demanda (%).

En consonancia con el desconocimiento de las propuestas metodológicas existentes, el 65,7% de empresas de la demanda no tienen intención de implantar ninguna metodología, estándar o norma oficial a corto plazo. Este porcentaje desciende a un 39,3% para las empresas de la oferta.

El modelo con un mayor grado de penetración en las pymes es CMMI [1] (7,9%), al igual que en la gran empresa, donde alcanza el 23,9%.

Las demás metodologías, modelos y estándares (ITIL [2], RUP, IEEE, etc.) no suponen un porcentaje representativo, de modo que en algunos casos no llegan a representar más del 1% del total de los resultados cuantitativos obtenidos.

3.1.2. Conocimiento de las herramientas y nivel de uso

Con el avance tecnológico, se ha ido incorporando al mercado una heterogénea gama de herramientas como instrumentos de apoyo a los proyectos de software. Su ámbito también se ha ido ampliando, proporcionando total cobertura a los servicios y al ciclo de vida del software.

Es significativo el predominio de las empresas orientadas a la oferta de servicios y productos de software que utilizan algún tipo de herramienta (61,4%) para ejecutar, al menos en parte, los procesos del ciclo de vida del software. El 22,5% de las empresas no utiliza ningún tipo de herramienta de soporte a los proyectos de software, o las usa parcialmente como apoyo a tareas y procesos específicos. Este dato pone de relieve que, en la actualidad, el uso de las herramientas ha llegado a convertirse en una necesidad para las empresas.

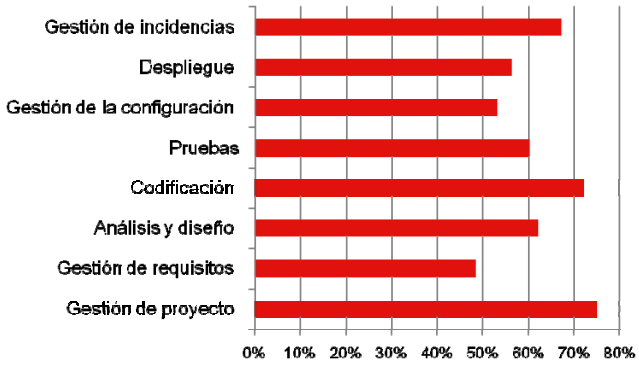


Figura 2. Uso de herramientas en las distintas fases del ciclo de vida de un proyecto de software (%).

Ante esta situación, se hace necesario que todos los agentes e instituciones relacionadas con la calidad del software se comprometan cada vez más para el establecimiento de acuerdos y políticas de actuación consensuadas que permitan la adopción progresiva de una cultura de la calidad, hasta llegar a su institucionalización como instrumento para alcanzar la mejora continua.

3.2. Recomendaciones

Se propone a continuación una serie de recomendaciones a los diferentes agentes implicados en el proceso de mejora de la calidad de los productos y servicios finales de software (empresas, profesionales y entidades públicas) a través de la difusión e implantación de las metodologías y herramientas aplicables a los procesos del ciclo de vida del software.

Respecto a las empresas, cabe destacar:

- La exigencia a los proveedores del seguimiento de alguna metodología y/o la utilización de alguna herramienta en los proyectos de software o, en su defecto, la exigencia de que su personal esté en posesión de alguna preparación y/o titulación específica, ya sea técnica (en herramientas) o metodológica.
- La certificación del software que asegure que cumple unos mínimos homologables y comunes a toda la industria.
- La alineación de las empresas demandantes con las mejores prácticas del gobierno de las tecnologías de la información que posibilite la definición e implantación de acuerdos de nivel de servicio con los proveedores.
- La realización de controles de calidad por parte de personas independientes al desarrollo que garanticen que el producto ha pasado previamente por diferentes tipos de pruebas que verifiquen que cumple con los requisitos acordados.
- La consolidación del uso de instrumentos que aseguren el procesamiento y análisis de la información relativa al avance del proyecto, como indicadores de proceso, de ejecución y/o de producto.

En cuanto a la administración pública, en su labor de tutelaje y coordinación, desde un punto de vista formativo y divulgativo, tenemos:

- El impulso de la formación continua de los profesionales de las empresas, dando una mayor difusión a los programas, seminarios y otros modos de formación relacionados con la calidad del software.
- El impulso de medidas que permitan instaurar una cultura de la calidad en las empresas españolas hasta alcanzar su institucionalización como instrumento válido de mejora de sus procesos internos, de su competitividad y de su productividad.

- La adopción de acciones para concienciar a los profesionales de los diferentes beneficios que la adopción de metodologías y herramientas aporta a su organización.
- La mayor participación de los organismos públicos de ámbito provincial y local.

Y atendiendo a un punto de vista ejecutivo y económico, destacamos:

- La inclusión en los concursos públicos, como mérito o requisito puntuable para las empresas, del hecho de estar certificadas o en trámite de certificación por alguna metodología formal y/o utilizar herramientas específicas en las fases del ciclo de vida del software.
- El incentivo fiscal a las empresas que se certifiquen y mantengan esa certificación durante el período de tiempo exigido por la ayuda o subvención en cuestión.
- El uso de las empresas destacadas en cada uno de los sectores de actividad económica como "tractores" o "ganchos" de las demás empresas, especialmente, de las pequeñas y medianas empresas.

Dirigidas a los centros de enseñanza, en particular, a la universidad:

- La difusión de la cultura de la calidad como parte del proceso de concienciación de los futuros profesionales de las TIC, organizando eventos conjuntos con los laboratorios y centros de referencia y/o estableciendo cursos de formación específica.
- El análisis de la posibilidad de incluir en los planes de estudio, especialmente en las carreras relacionadas con la informática, materias relacionadas con la calidad del software alineándolas con las necesidades reales del mercado.
- La participación en programas que apoyen la adopción de metodologías y herramientas en las empresas españolas, especialmente orientados a las pequeñas y medianas empresas.

A pesar de este objetivo común, no fácil establecer acuerdos y políticas consensuadas de actuación, por estar presentes diferentes intereses. Por ello, desde la perspectiva de los diferentes agentes afectados, se reclama la presencia de una figura que coordine las diversas medidas y aúne esfuerzos para, así, establecer un marco común de actuación para todas las partes involucradas.

4. Estudio sobre el modelo de factorías de software con un enfoque *nearshore*

El trabajo pretende articular un conocimiento profundo sobre el fenómeno de las factorías de software en España y proponer aspectos clave para mejorar el posicionamiento de esta industria en el exterior. En definitiva, se trata de intentar responder a la pregunta de si España se puede considerar un destino *nearshore* para factorías de software y, en este contexto, qué papel pueden desempeñar las empresas y las administraciones públicas para sostener el modelo de factorías de software con un enfoque *nearshore*.

Para llevar a cabo la investigación se ha utilizado una metodología cualitativa centrada en el análisis en profundidad de los casos de éxito detectados y el contraste de información con estudios nacionales e internacionales disponibles. Se dirigieron 23 entrevistas a directores de factorías de software, a los directores de las áreas de industria y/o promoción industrial (empresarial) de las comunidades autónomas en las que están actuando las factorías y a otros informantes críticos identificados a lo largo del estudio.

4.1. Diagnóstico

Se ha realizado un análisis de las debilidades, amenazas, fortalezas y oportunidades de España como destino *nearshore*.

Si en España se opta por competir en el entorno internacional con un enfoque basado en los modelos tradicionales (mano de obra intensiva a bajo coste), se evidencian serios riesgos para la sostenibilidad del modelo en el futuro. En cambio, si en España se opta por articular un modelo basado en ventajas competitivas específicas, se proyecta un sólido posicionamiento como destino *nearshore* [3].

Independientemente de las ventajas con las que cuenta España en infraestructura de comunicaciones, calidad de vida, alineamiento económico y político con Europa..., resalta la experiencia y consolidación de los desarrollos en sectores y productos de valor en el mercado del software. Nos referimos al *know how* español en software dirigido a sectores como banca, seguros, turismo, etc. Una estrategia de consolidación y proyección del software español en dominios concretos y especializados puede ser más competitiva en Europa y más fácilmente sostenible en el tiempo. En esta misma línea está trabajando la Asociación de Empresas Españolas de Consultoría (AEC), que ha definido el modelo Value Shore España, articulado a partir del reconocimiento de que en España existen

determinados servicios y desarrollos que están suficientemente maduros, probados, validados y que añaden valor al cliente. En definitiva, España cuenta con una serie de ventajas para posicionarse como destino *nearshore* en un modelo intensivo en conocimiento, más que en mano de obra exclusivamente.

Debilidades	**Amenazas**	**Fortalezas**	**Oportunidades**
• Tamaño reducido de las factorías. • Tendencia al *body shopping* más que a la estandarización de productos y metodología. • Insuficiente calidad en los procesos. • Falta de certificaciones. • Poca capacidad de exportación de software. • Escasez de recursos humanos con formación especializada y con idiomas (inglés). • Falta de sincronía entre los grados de industrialización del *Front Office* y *Back-Office.* • Falta de políticas públicas orientadas específicamente a potenciar las factorías de software en cada territorio. • Falta de suelo industrial. • Falta de criterios homogéneos sobre lo que es una factoría de software.	• Anclaje cultural por parte de determinados clientes que los limita a trabajar en un proceso de producción *nearshore*. • Competencia de otros países y regiones europeas. • Aumento de la escasez de profesionales frente a la alta demanda en el mercado. • Dificultad para mantener tarifas competitivas ante la escasez de recursos humanos. • Aumento de la competencia por los recursos humanos dentro de España o dentro de una misma provincia. • Relajación en los estándares y modelos de certificación. • Traslado a los desarrollos a otros países *Offshore.*	• Buenas comunicaciones. • Buena infraestructura de telecomunicaciones. • Buenas universidades en diferentes provincias españolas con carreras TIC. • Probada experiencia de las empresas españolas en determinados sectores y talento de sus profesionales. • Cercanía al cliente final. • En algunas factorías y sectores se cuenta con metodologías maduras y procesos de gestión adecuados. • Estructura salarial competitiva en comparación con otros países. • Alineamiento político y económico con Europa. • Política activa en materia de I+D+i. • Territorio con alta calidad de vida.	• Posicionamiento en el entorno internacional como destino atractivo para FSW. • Avance en un proceso de madurez en la industria del software: mejora de costes y de productividad. • Mano de obra que quiere quedarse en su tierra, con lo cual disminuye la rotación de personal. Ayudas de las administraciones públicas. • Buenas relaciones con la administración pública para potenciar marcas. • Consolidación de las empresas matrices que han generado factorías de software.

Tabla 3. Análisis DAFO.

4.2. Recomendaciones

Las recomendaciones que a continuación se plantean están orientadas a proporcionar mayor solidez a una estrategia *nearshore* basada en capitalizar la experiencia española en productos sofisticados, de calidad y que agregan valor.

En cuanto a las factorías de software, destacamos:

- Diferenciar la producción de software por valor, introduciendo productos sectoriales de alta calidad en el mercado del software.
- Reorientar el enfoque de la producción hacia productos y paquetes parametrizables de alto valor.
- Avanzar en el proceso de certificación, aprovechando las ayudas que para tal efecto ofrece INTECO, a través de su línea de calidad del software.
- Reclutar y/o atraer recursos humanos a las factorías de software desarrollando acuerdos efectivos con los centros de formación profesional con carreras TIC para alinear la formación de los centros con las necesidades de las factorías o atrayendo a profesionales de terceros países, aprovechando la iniciativa europea de la "tarjeta azul".

Para las administraciones autonómicas:

- Contar con planes específicamente dirigidos a potenciar las factorías de software en sus territorios mediante la realización de diagnósticos detallados del territorio para atraer a las factorías de software, la gestión de suelo urbano para uso industrial, el tutelaje y el acompañamiento a la creación de factorías.

Respecto a la administración general del estado, tenemos:

- Crear un plan de definición de criterios comunes y de ordenamiento del sector (condiciones que debe cumplir una factoría, quién puede crear una factoría, principios de calidad...).

En cuanto a las universidades, destacamos:

- Mejorar la formación en inglés de los recursos humanos españoles, ya que el inglés es la lengua de uso común en el mercado del software.
- Coordinarse con las factorías de software en favor de la mejor y más pertinente formación del alumnado de carreras TIC para hacerla coherente con las necesidades del mercado. Esta coordinación se materializa, entre otras cosas, en el desarrollo de asignaturas de los últimos cursos íntegramente impartidas por las factorías de software o en la realización de proyectos de fin de carrera en las factorías.

Finalmente, dirigidas a la Asociación Española de Empresas de Consultoría:

- Elaborar una estrategia del sector para la negociación con el Ministerio de Trabajo e Inmigración sobre cupos para trabajadores extranjeros cualificados en TIC.
- Identificar universidades latinoamericanas y de países del Este, para establecer acuerdos con el fin de captar talento y de formar en origen con el compromiso de contratación en empresas españolas.
- Coordinarse con las ONG y asociaciones de inmigrantes para gestionar los procesos de captación de recursos humanos cualificados de terceros países, velando así por una estrategia sostenible en la que "todos ganen".

5. Conclusiones

El diagnóstico forma parte del planteamiento metodológico de INTECO, ya que sobre su base se asienta el diseño de los servicios prestados.

Los estudios presentados en este artículo pretenden mostrar cuál es la situación de la calidad del software en la industria española, así como proporcionar unas recomendaciones de actuación a los distintos agentes involucrados en este ámbito.

Los resultados de estos trabajos estarán disponibles, próximamente, de forma más amplia y en un formato accesible, en la sección de calidad del software de INTECO www.inteco.es.

Agradecimientos

INTECO agradece su colaboración a todos los agentes que han participado en los estudios que aquí se presentan, por la aportación de sus experiencias y conocimientos. Ellos han hecho posible la elaboración de un diagnóstico de la situación actual de la industria española en el ámbito de la calidad del software.

Referencias

[1] Chrissis, M. B., Konrad, M. y Shrum, S., *CMMI® Guidelines for Process Integration and Product Improvement,* Addison-Wesley, 2006.

[2] OGC (Office of Government Commerce), *ITIL Lifecycle Publication Suite. Version 3,* OGC, 2007.

[3] Parveen, K., Marriott, I., Hallawell, A. y Scardino, L., *Analysis of Spain as an Offshore Services Location. ID Number: G00152674,* Gartner, 2007.

Revista Española de Innovación, Calidad e Ingeniería del Software, Vol.4, No. 2, 2008

Accesibilidad web: un vistazo a tres webs de administraciones públicas en España

Jorge Sánchez, Tanja E. J. Vos
Instituto Tecnológico de Informática. Universidad Politécnica de Valencia
{jordisan, tanja}@iti.upv.es

Abstract

Spanish law dictates that the web site of a public agency must reach a specific level of accessibility. However, achieving effective accessibility for a web site is not an easy task. In this paper, we will check the actual status of accessibility of some public administration web sites by testing a small sample. Doing this, we will experience how complicated it is to reach and/or verify the compliance of accessibility standards.

Key words: web accessibility, regulation, UNE, WAI, WCAG, public agencies.

Resumen

La legislación española obliga a que los sitios web de las administraciones públicas alcancen un determinado nivel de accesibilidad; pero conseguir que una web sea realmente accesible es una tarea complicada. Con una pequeña muestra vamos a estudiar el estado real de la accesibilidad de algunos de esos sitios, al mismo tiempo que comprobamos hasta qué punto es complejo alcanzar y/o verificar el cumplimiento de los estándares en accesibilidad.

Palabras clave: accesibilidad web, normativa, UNE, WAI, WCAG, administraciones públicas.

1. Introducción

Uno de los objetivos de la tecnología web es ser accesible para el mayor número posible de usuarios. En el caso concreto de España, desde la publicación en el Boletín Oficial del Estado (BOE) del Real Decreto 1494/2007 [1], las páginas en Internet de las administraciones públicas deben cumplir una norma específica de accesibilidad.

Pero conseguir que, de una manera efectiva, el mayor número posible de personas pueda acceder a los contenidos de una aplicación o página web no es tarea fácil. Así pues, ¿cuál es el estado real, en cuanto a accesibilidad, de las páginas de las administraciones públicas? El objetivo de este artículo no es realizar un estudio detallado de dichos sitios, sino hacer una breve inspección de algunas páginas para comprobar cuál es su nivel de accesibilidad desde un punto de vista fundamentalmente práctico, así como obtener una

idea de la complejidad y el alcance real de este tipo de evaluaciones. Para empezar, introduciremos algunos conceptos básicos.

1.1. ¿Qué es accesibilidad web?

Según el *World Wide Web Consortium* (conocido como W3C: http://www.w3.org/), el principal organismo estandarizador en Internet, "hablar de accesibilidad web es hablar de un acceso universal a la web, independientemente del tipo de hardware, software, infraestructura de red, idioma, cultura, localización geográfica y capacidades de los usuarios".

Como se deduce de esta afirmación, y en contra de lo que se cree habitualmente, las cuestiones de accesibilidad no se refieren únicamente a los usuarios con algún tipo de discapacidad (física o intelectual), sino que se dirige a cualquier clase de problema que impida el acceso a los contenidos de la web. Son problemas de accesibilidad los que tiene una persona ciega para acceder a una página web, pero también lo son los que tiene un usuario de edad avanzada, alguien con una conexión especialmente lenta o que usa un software anticuado, una persona que no domina el idioma de la página, o los que tiene alguien que esté accediendo desde un dispositivo móvil.

De esto se deriva que no exista una "accesibilidad absoluta" como tal; el grado de accesibilidad de una página dependerá de qué usuarios puedan utilizarla y de las dificultades que tengan para hacerlo.

Dado que, en la práctica, resulta imposible comprobar hasta qué punto una página resulta accesible para todas las posibles combinaciones de usuarios y situaciones, se suele decir que "una página o sitio web es accesible" cuando se quiere indicar que cumple un conjunto de criterios y estándares de accesibilidad generalmente aceptados y conocidos.

1.2. Estándares y legislación

La principal referencia en cuanto a estándares de accesibilidad web proviene del W3C, en concreto, de su *Web Accessibility Initiative* (WAI) [2]; sus recomendaciones, como otras del W3C, constituyen el estándar de facto en tecnología web.

WAI publica diferentes tipos de directrices dirigidas a desarrolladores de navegadores, de herramientas de edición, etc. Las que afectan a los contenidos web son las Web Content Accessibility Guidelines (WCAG) [3], cuya versión más reciente es la 1.0, que será próximamente reemplazada por la 2.0.

Las directrices de WCAG constituyen unas recomendaciones generales sobre las características que debe cumplir el contenido web para ser accesible en la mayor medida posible (haciendo referencia al lenguaje utilizado, a los colores, a las tablas, a las tecnologías utilizadas, etc.). A su vez, las directrices incluyen puntos de verificación o control (checkpoints) más específicos con tres posibles niveles de prioridad en función de que se consideren más o menos importantes. Así, se dice que las páginas alcanzan el nivel A de accesibilidad cuando cumplen todos los puntos de control de prioridad 1; tienen el nivel AA cuando cumplen los de prioridad 1 y 2; y el nivel AAA cuando cumplen los de prioridad 1, 2 y 3.

En España, la normativa sobre accesibilidad web de las administraciones públicas a la que nos referíamos al principio especifica que sus páginas deben cumplir, como mínimo, los niveles 1 y 2 de la norma UNE 139803:2004 [4]. Esta norma es una adaptación española de la WCAG 1.0 que resulta casi equivalente a ella; de hecho, la propia norma incluye un anexo de correspondencia entre ambas normativas.

No existen organismos que oficialmente certifiquen la accesibilidad de una página web de un modo absoluto. Son los propios responsables de las páginas web los que deciden incluir en ellas un certificado (normalmente, en forma de sello) o declaración de cumplimiento de las directrices, que indican más un compromiso que una garantía de ser realmente accesibles, ya que ellos mismos son quienes comprueban dicha accesibilidad.

Algunas organizaciones como AENOR (en el caso de las UNE: http://www.accesible.aenor.es/index.asp?MP=2&MS=21&MN=1) o Technosite (en el caso de WCAG: http://www.technosite.es/auditoriaycertificacion.asp) ofrecen servicios de pago de certificación de accesibilidad, pero sus certificados no son obligatorios para los sitios web que pretendan cumplir una u otra normativa.

1.3. Evaluaciones de accesibilidad

Una evaluación de accesibilidad (o de conformidad) determina si un sitio web cumple con los estándares de accesibilidad (como los WCAG). Este tipo de evaluaciones suele realizarse mediante un método [6] que incluye testeo automático, semiautomático y manual. Consta de los siguientes pasos:

1. Determinar el alcance de la evaluación. Se trata de definir el nivel de accesibilidad que se desea alcanzar, así como seleccionar una muestra de páginas representativas

del sitio que se va a evaluar (en la práctica, resulta imposible evaluar la accesibilidad de cada una de las páginas del sitio web, a no ser que se trate de un sitio de tamaño muy reducido).

2. Utilizar herramientas automáticas de evaluación. Hay que validar de modo automático que los lenguajes utilizados (HTML, CSS, etc.) cumplen con su gramática formal (es decir, que están "bien construidos") y hay que utilizar herramientas automáticas de evaluación de accesibilidad; en este caso, se aconseja usar, al menos, dos de esas herramientas.
3. Evaluar manualmente las páginas representativas. Dado que los principios de accesibilidad son interpretables, no es suficiente con la validación automática y es necesario que expertos en accesibilidad revisen las páginas representativas del sitio web en busca de problemas de accesibilidad. Estos se apoyan en técnicas (como listas de puntos de control) que deben cumplir las páginas, la utilización de navegadores gráficos en diferentes plataformas y con diferentes configuraciones (desactivando JavaScript, CSS, imágenes…), el uso de navegadores especializados (solo texto, lectores de pantalla)...
4. Resumen de resultados. Se ha de elaborar un informe con las conclusiones obtenidas, incluyendo los problemas de accesibilidad detectados y las recomendaciones para resolverlos. También es posible incluir sesiones con usuarios reales en el testeo de accesibilidad [7], pero esto solo se suele hacer en casos muy específicos y de forma complementaria, ya que resulta imposible hacer un testeo real con toda la tipología de accesos al sitio web: discapacidades, dispositivos, configuraciones, etc.

2. Análisis de algunos sitios web de las administraciones públicas

A la vista de lo que se ha señalado, cabe preguntarse en qué situación se encuentran las webs de las administraciones públicas en España. Para responder, hemos realizado un breve análisis de algunas de esas páginas; ni el muestreo ni el análisis realizado pretenden ser exhaustivos, sino proporcionar una idea general de a situación y comprobar el grado de dificultad que entraña conseguir y verificar que un sitio web es accesible en la práctica.

Las páginas seleccionadas han sido las iniciales de los sitios web oficiales de estos organismos, algunas de cuyas características las hacen interesantes para este estudio:

Dirección General de la Policía y la Guardia Civil. Consideramos que es una página especialmente relevante, ya que sus usuarios potenciales son la práctica totalidad de la población, al incluir trámites tan importantes y usuales como la renovación de documentos identificativos, la presentación de denuncias, etc.

Ministerio de Fomento. Este portal fue galardonado con un premio por su accesibilidad, así que resulta un buen candidato al análisis para comprobar hasta qué punto es realmente accesible un sitio web de esas características.

Ayuntamiento de Valencia. Esta web cuida especialmente la accesibilidad y toma como muestra de una administración pública no estatal (en este caso, municipal). Además, resulta interesante su implementación de una "versión accesible".

Dado que no pretendemos hacer un análisis exhaustivo de la accesibilidad de estos sitios, sino simplemente extraer algunas conclusiones de interés, hemos realizado un subconjunto de las tareas propias de una evaluación de accesibilidad:

1. Determinación de la URL (dirección) de la página inicial del sitio web.
2. Comprobación de las declaraciones de accesibilidad de la propia página: sellos y páginas específicas.
3. Validación automática con dos herramientas: *PISTA* (para la norma UNE hasta nivel 2: http://www.mityc.es/Pista/Contenido/AccesibilidadFreeware/) y *Cynthia Says* (para la norma WCAG hasta nivel 2: http://www.contentquality.com). Se recoge el número de errores detectados de cada prioridad.
4. Validación manual de algunos puntos de control.
5. Comentarios y conclusiones sobre la web.

2.1 Dirección General de la Policía y la Guardia Civil

2.1.1. URL

http://www.policia.es/. Se trata de una página de marcos (frames); no se revisan las páginas interiores de los marcos.

2.1.2. Declaraciones de accesibilidad

La página no hace ninguna referencia a certificaciones, ni incluye ningún enlace ni información explícita sobre accesibilidad.

Figura 1. Web de la Dirección General de la Policía y la Guardia Civil.

2.1.3. Validación automática

Con la herramienta PISTA (UNE 139803):

- **Tres errores de prioridad 1** (además, señala 7 comprobaciones manuales). Entre los errores detectados, no se indica el idioma principal de la página. Además, al tratarse de una página de marcos, debería asignar un título a cada uno de los marcos para facilitar la navegación entre ellos.
- **Tres errores de prioridad 2** (además, señala 15 comprobaciones manuales). Los errores son consecuencia de que el código fuente incluye atributos no válidos según la gramática declarada por la página.

Con la herramienta Cynthia Says (WCAG):

- **Un error de prioridad 1.** Al igual que con PISTA, se detecta que los marcos no tienen un título que los identifique.
- U**n error de prioridad 2.** Para esta prioridad, los marcos deberían incluir también (y no lo hacen) una descripción textual de su contenido.

2.1.4. Validación manual

La validación manual sobre una página de marcos tiene un alcance limitado si no se acompaña de la validación sobre las páginas interiores de cada marco. Aun así, se detectan algunos problemas de accesibilidad: por ejemplo, la página de marcos no proporciona un contenido alternativo para navegadores que no soportan el uso de marcos.

2.1.5. Comentarios

Las páginas de marcos (frames) son especialmente problemáticas a la hora de conseguir que sean accesibles [8]. Por tanto, deben evitarse para conseguir una mayor accesibilidad o, al menos, en el caso de utilizarse, es importante ser muy escrupulosos con los estándares de accesibilidad que hacen referencia a ellas.

En esta página en concreto, el hecho de que los marcos no incluyan un título o una descripción de su contenido, provoca que la navegación entre ellos sea especialmente problemática, por ejemplo, para los usuarios que utilizan lectores de pantalla.

En general, no da la impresión de que en esta página se hayan realizado esfuerzos explícitamente dirigidos a conseguir una mayor accesibilidad de sus contenidos.

2.2. Ministerio de Fomento

2.2.1. URL

http://www.fomento.es/mfom/lang_castellano/default.htm/. URL de la página inicial en castellano, redirigida desde http://www.fomento.es/.

Figura 2. Web del Ministerio de Fomento.

2.2.2. Declaraciones de accesibilidad

La página no hace referencia a la norma UNE. Incluye sellos que declaran que la página cumple con WCAG, nivel de conformidad AA, incluyendo el sello estándar de WAI y también el de Technosite .

El enlace Accesibilidad, en la zona inferior, lleva a una página sobre accesibilidad del sitio web en la que se explica la conformidad de la página con las normas WCAG; además, incluye un enlace para acceder a una versión del sitio web de "alto contraste".

2.2.3. Validación automática

Con PISTA (UNE 139803):

- **Cero errores de prioridad 1** (señala 57 comprobaciones manuales). Las comprobaciones manuales incluyen, entre otras, la revisión de los textos alternativos a las imágenes que aparecen en la página.
- **Siete errores de prioridad 2** (además, señala 84 comprobaciones manuales). En concreto, el código fuente incluye caracteres no válidos, lo que hace que no valide la gramática declarada. Además, la herramienta parece detectar un "falso positivo", ya que señala un elemento como desaconsejado, cuando en realidad es una declaración de clase válida (<p class="alignCenter">).

Con Cynthia Says (WCAG):

- **Cero errores de prioridad 1.**
- **Tres errores de prioridad 2.** Uno de los errores lo produce un mismo texto ("benvinguts") enlazado dos veces, pero con diferentes destinos en cada caso (versiones en catalán y valenciano), lo cual está desaconsejado.

2.2.4. Validación manual

Algunos de los problemas de accesibilidad detectados en una validación manual son:

Las imágenes de las noticias de portada no incluyen texto alternativo (está en blanco). Se trata de fotografías de actos oficiales que no se pueden considerar imágenes decorativas y que deberían ir acompañadas de una descripción textual. Este punto tiene prioridad 1 tanto en las normas UNE como en las WCAG.

El menú de la izquierda ofrece poco contraste de color con el fondo (se puede verificar con alguna herramienta online: http://www.accesskeys.org/tools/color-contrast.html). Este punto tiene prioridad 2 tanto en la norma UNE como en las WCAG. Es

cierto que existe una versión de alto contraste del sitio web enlazada desde la página de declaración de accesibilidad, pero la página que estamos evaluando no cumple este punto.

2.2.5. Comentarios

No cabe duda de que en este sitio sí se han realizado esfuerzos para conseguir que la página sea accesible, incluyendo certificaciones de organizaciones externas; incluso declaran haber obtenido en 2007 un premio a la web pública más accesible (premios TAW: http://www.tawdis.net/taw3/cms/es/premio/premiados.html).

No obstante, se detectan algunos problemas de accesibilidad. Los relativos a las validaciones automáticas son poco importantes (o incluso inexistentes, como en el caso del "falso positivo"); importan más los problemas de contraste de color y de ausencia de descripciones textuales de imágenes que se comprueban con una validación manual.

Es posible que la ausencia de alternativas textuales en las imágenes sea más responsabilidad del editor de contenidos (simplemente, no ha incluido la descripción) que de la propia implementación técnica del sitio web. Esto evidencia la importancia de revisar la accesibilidad de modo periódico, ya que es una característica que tiende a degradarse con el paso del tiempo, a medida que se realizan modificaciones.

Más llamativo resulta el problema del contraste en los textos, en especial porque resultaría relativamente sencillo corregirlo en la versión "normal" del sitio web, sin necesidad de incluir una versión alternativa de alto contraste (que, además, los usuarios tienen que localizar, tarea que no es trivial).

2.3. Ayuntamiento de Valencia

2.3.1. URL

http://www.valencia.es/ayuntamiento2/ndportada.nsf/(Portadas1)/$first?opendocument&lang=1. Versión en castellano de la página inicial; el idioma se selecciona en http://www.valencia.es/.

2.3.2. Declaraciones de accesibilidad

No hace referencia a la norma UNE. Incluye un sello de cumplimiento WCAG AA. El icono superior derecho da paso a una versión "accesible" de la página, que consiste en un breve texto explicativo y en enlaces a las principales opciones de menú del sitio web. La página incluye el texto "Optimizado 800x600" (resolución de pantalla).

Figura 3. Web del Ayuntamiento de Valencia.

2.3.3. Validación automática

Con PISTA (UNE 139803):

- **Cero errores de prioridad 1** (señala 80 comprobaciones manuales).
- **Un error de prioridad 2** (además, señala 36 comprobaciones manuales). El error detectado es bastante habitual [9] y se debe a un uso no correcto del carácter & en el código fuente, lo que provoca que no se valide la gramática declarada. A pesar de esto, normalmente los navegadores lo interpretan correctamente y en la práctica no causa problemas.

Con Cynthia Says (WCAG AA):

- **Cero errores de prioridad 1.**
- **Un error de prioridad 2.** No se indica el idioma del texto con una técnica en concreto (etiqueta <meta>), aunque sí se hace mediante otros mecanismos.

2.3.4. Validación manual

Algunos problemas detectados con una validación manual son:

- Los menús desplegables de la izquierda, programados con JavaScript, impiden que puedan recorrerse de modo secuencial mediante el teclado (al menos, en Firefox versión 2.0 bajo Windows).

El hecho de que el sitio esté optimizado para una determinada resolución (800x600) no debería impedir que se usase desde plataformas con resoluciones diferentes (incluyendo dispositivos móviles, con pantallas más pequeñas). En este sentido, sería necesario hacer una comprobación más detallada con diferentes dispositivos, navegadores, configuraciones, etc.

En la zona derecha conviven hasta ocho animaciones diferentes que pueden resultar un problema, por ejemplo, para usuarios con problemas de atención. Aunque no se haya detectado por los validadores automáticos, WAI recomienda evitar animaciones en las páginas mientras los navegadores no permitan a los usuarios detenerlas [10].

2.3.5. Comentarios

La página tiene un alto nivel de accesibilidad, siendo evidente el esfuerzo que se ha realizado para conseguirlo; los errores detectados con los validadores automáticos son poco importantes. No obstante, vemos que una validación manual, aunque no sea excesivamente detallada, sí detecta algunos posibles problemas.

Resulta llamativo que exista una "versión accesible" diferenciada, cuando los contenidos de la versión estándar ya alcanzan un alto nivel de accesibilidad. Esta versión "simplificada" del porta podría ser de ayuda a un determinado tipo de usuarios, a pesar de que WCAG recomienda utilizar una versión alternativa únicamente cuando no sea posible conseguir una versión estándar accesible [11].

3. Conclusiones

Las páginas de las administraciones públicas ofrecen diferentes estados en cuanto a accesibilidad, aunque se aprecia un esfuerzo en ese sentido. Curiosamente, ninguna de las páginas analizadas hace referencia aún a la norma UNE 139803:2004, pues todavía tienen como referente la WCAG 1.0; cabe preguntarse qué ocurrirá en el momento en que se publiquen como definitivas las WCAG 2.0. Hemos comprobado también que la inclusión de sellos y declaraciones de accesibilidad no garantizan que el sitio web cumpla los

criterios señalados, aunque sí son una señal de que los responsables del sitio han realizado un esfuerzo importante para conseguir que sea accesible.

Los ejemplos analizados muestran que la validación de accesibilidad de un sitio web (¡aunque sea solo de la página inicial!) no se consigue únicamente con la utilización de validadores automáticos, ya que pueden detectar problemas poco importantes (o incluso, inexistentes), ignorando problemas potencialmente más importantes que solo se pueden detectar mediante una comprobación manual realizada por expertos.

Por último, es necesario recordar que las comprobaciones de este artículo se han limitado a la página inicial. En cuanto a las páginas interiores de estos sitios, probablemente estén más descuidadas en este aspecto, ya que las revisiones de accesibilidad se suelen empezar por la página inicial, a pesar de que es posible que la accesibilidad de una página interior (por ejemplo, la realización de un trámite importante) sea más necesaria para el usuario que la de la página inicial.

Referencias

[1] *Boletín Oficial del Estado 279 de 21/11/2007*, sec. 1, pp. 47.567-47.572.

[2] W3C, *Web Accessibility Initiative (WAI).* http://www.w3.org/WAI/ (consultada en julio de 2008)

[4] W3C, *Web Content Accessibility Guidelines 1.0.* http://www.w3.org/TR/WCAG10/ (consultada en julio de 2008)

[5] AENOR, *Norma UNE 139803:2004*, AENOR, 2004.

[6] W3C, *Conformance Evaluation of Web Sites for Accessibility.* http://www.w3.org/WAI/eval/conformance.html (consultada en julio de 2008)

[7] W3C, *Involving Users in Web Accessibility Evaluation.* http://www.w3.org/WAI/eval/users.html (consultada en julio de 2008)

[8] W3C, *HTML Techniques for Web Content Accessibility Guidelines 1.0: Frames.* http://www.w3.org/TR/WCAG10-HTML-TECHS/#frames (consultada en julio de 2008)

[9] *Common HTML Validation Problems: Ampersands (&'s) in URLs.* http://htmlhelp.com/tools/validator/problems.html#amp (consultada en julio de 2008)

[10] W3C, *Techniques for Web Content Accessibility Guidelines 1.0: Checkpoint 7.3.* http://www.w3.org/TR/WCAG10-TECHS/#tech-avoid-movement (consultada en julio de 2008)

[11] W3C, *Techniques for Web Content Accessibility Guidelines 1.0: Checkpoint 11.4.* http://www.w3.org/TR/WCAG10-TECHS/#tech-alt-pages (consultada en julio de 2008)

Revista Española de Innovación, Calidad e Ingeniería del Software, Vol.4, No. 2, 2008

Infraestructura de pruebas para una plataforma de inteligencia de negocios: lecciones aprendidas de una experiencia académica

Ruth Alarcón, Carla Basurto, Abraham Dávila
Departamento de Ingeniería. Pontificia Universidad Católica del Perú
{ruth.alarcong, cbasurto, abraham.davila}@pucp.edu.pe

Abstract

The software testing is a process that is regularly conformed by repetitive tasks, that is why automating them becomes attractive and convenient in different situations. In our country many enterprises develop software with adjusted schedules, not considering enough time to perform tests adequately. A project for developing a platform for business intelligence should consider test automation and the implementation of a testing infrastructure of the platform which will have many versions. This paper contains some experiences about the implementation of a test infrastructure for business intelligence platform; both were developed in academic and business environments.

Key words: software testing, web testing, functional testing, business intelligence, automation software testing.

Resumen

La prueba de software es un proceso que, por lo general, tiene tareas que se repiten y cuya automatización resulta atractiva y conveniente en diversas situaciones. En nuestro medio, muchas empresas desarrollan software a medida con plazos ajustados que no les permite realizar pruebas de manera adecuada. La construcción de una plataforma de inteligencia de negocios es un proyecto en el que resulta muy conveniente automatizar las pruebas y la implementación de una infraestructura de pruebas de la plataforma que tendrá sucesivas versiones. Este artículo recoge algunas experiencias surgidas al implementar una infraestructura de pruebas para una plataforma de inteligencia de negocios, desarrolladas ambas en entornos académicos y empresariales.

Palabras clave: pruebas de software, pruebas en web, pruebas funcionales, inteligencia de negocios, automatización de pruebas de software.

1. Introducción

La prueba de software, según IEEE [1], es una actividad en la que un sistema o componente se ejecuta bajo condiciones específicas, se observa o registran los resultados y se realiza

una evaluación de un aspecto del sistema o componente. Para el SWEBOK [2], se trata de una actividad que se realiza para evaluar y mejorar la calidad del producto a través de la identificación de defectos y problemas. En general, podemos decir que las pruebas contribuyen a obtener un mejor producto o un producto con menos defectos.

A partir de lo presentado por varios autores, como Presmman [3], Sommerville [4] y Futrell [5], se entiende que las pruebas se realizan en las etapas finales de cada iteración, o incluso a finales del proyecto; esto es casi independiente del ciclo de vida de construcción del software. Asimismo, muchas organizaciones que desarrollan software a medida no cuentan con tiempo suficiente para poder ejecutar las pruebas de manera adecuada [6]; esto provoca que los productos tengan una cantidad significativa de defectos que salen a la luz cuando el software es utilizado por los usuarios. El desarrollo de un producto de software a nivel comercial, para una gran cantidad de usuarios y con varias evoluciones en el futuro, suele seguir el mismo proceso que cualquier otro proyecto de desarrollo de software; sin embargo, dada su orientación de producto comercial, es razonable pensar que habrá más interés en la automatización de las pruebas con respecto a otros proyectos, como el desarrollo de software a medida.

Por otro lado, un grupo de profesores y estudiantes, con el apoyo de la empresa ACKLIS SAC, que cuenta con el apoyo de la Universidad, desarrolló una plataforma de inteligencia de negocios [7] en dos esfuerzos sucesivos. En el primer esfuerzo del proyecto se desarrolló la versión en cliente-servidor de la plataforma y en un segundo esfuerzo, la versión para web. Durante este segundo esfuerzo se planteó la necesidad de contar con una infraestructura de pruebas que permitiera apoyar el trabajo de todo el equipo, sobre todo de cara a las futuras evoluciones del producto.

En este artículo se presentan las experiencias recogidas durante la construcción de una infraestructura de pruebas de software que se ha desarrollado para una plataforma de inteligencia de negocios. La infraestructura permitirá realizar pruebas de regresión con casos de prueba funcionales y pruebas de carga. Tanto la plataforma como la infraestructura han sido desarrolladas por un equipo de estudiantes y profesores bajo el esquema de dos proyectos separados.

El trabajo se organiza así: en el segundo epígrafe se describe la plataforma de inteligencia de negocios; en el tercero se describe la infraestructura de pruebas desarrollada;

en el cuarto se recogen algunas buenas y malas prácticas encontradas en el desarrollo de la plataforma y que impactaron en el desarrollo de la infraestructura; finalmente, se presenta una discusión final sobre el tema.

2. BUFEO, la plataforma de inteligencia de negocios

La plataforma de inteligencia de negocios BUFEO es un proyecto que se inició en el año 2003 y se desarrolló en varias fases y cada fase, en varias etapas. La primera fase consistió en desarrollar una primera versión sobre una arquitectura cliente-servidor para permitir que el equipo de desarrollo se familiarizase con los aspectos funcionales de la inteligencia de negocios y técnicos de la programación. El proyecto se dividió en tres frentes de trabajo para cubrir los módulos de análisis, extracción y explotación. La integración de los módulos se realizó principalmente a nivel de archivos, pues los procesos no dependían entre sí. La segunda fase consistió en desarrollar la versión para una arquitectura web; en este caso se planteó un proyecto integrado desde el principio, teniendo en cuenta el conocimiento adquirido sobre inteligencia de negocios y la parte técnica. La segunda fase contó con el apoyo parcial del fondo del programa PROCOM del Consejo Nacional de Ciencia y Tecnología del Perú (CONCYTEC) (http://www.acklis.com/bufeo/).

La plataforma cubre los servicios propios de inteligencia de negocios desde la extracción, transformación y carga de datos al cubo de análisis, así como su posterior utilización (explotación de datos) a través de reportes y consolidados a diversos niveles; incluye herramientas de modelado y análisis, no siendo necesarios otros productos para que una empresa pueda utilizar inteligencia de negocios a nivel básico. La empresa, al utilizar BUFEO, se tiene que preocupar de realizar el modelado de negocios, las reglas de transformación y las salidas (reportes) en la plataforma.

La plataforma BUFEO está dominada principalmente por los tres componentes funcionales definidos al principio: análisis, extracción y explotación. En la Figura 1 se presentan, además de los componentes principales, los componentes control, gráficos y común que han sido desarrollados por el proyecto y otros de uso libre. En la figura 2 se muestran algunas clases de análisis, en especial de empresas por su carácter multiempresarial; proyecto para las distintas iniciativas de inteligencia de negocios de una

organización; tema, tablas (hechos y dimensiones) y campos para la definición de los almacenes de datos (Data Warehouse).

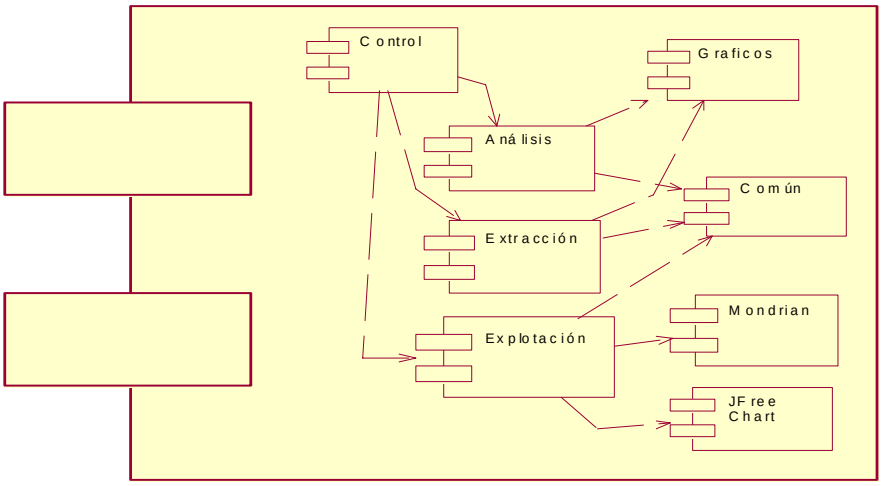


Figura 1. Arquitectura de BUFEO.

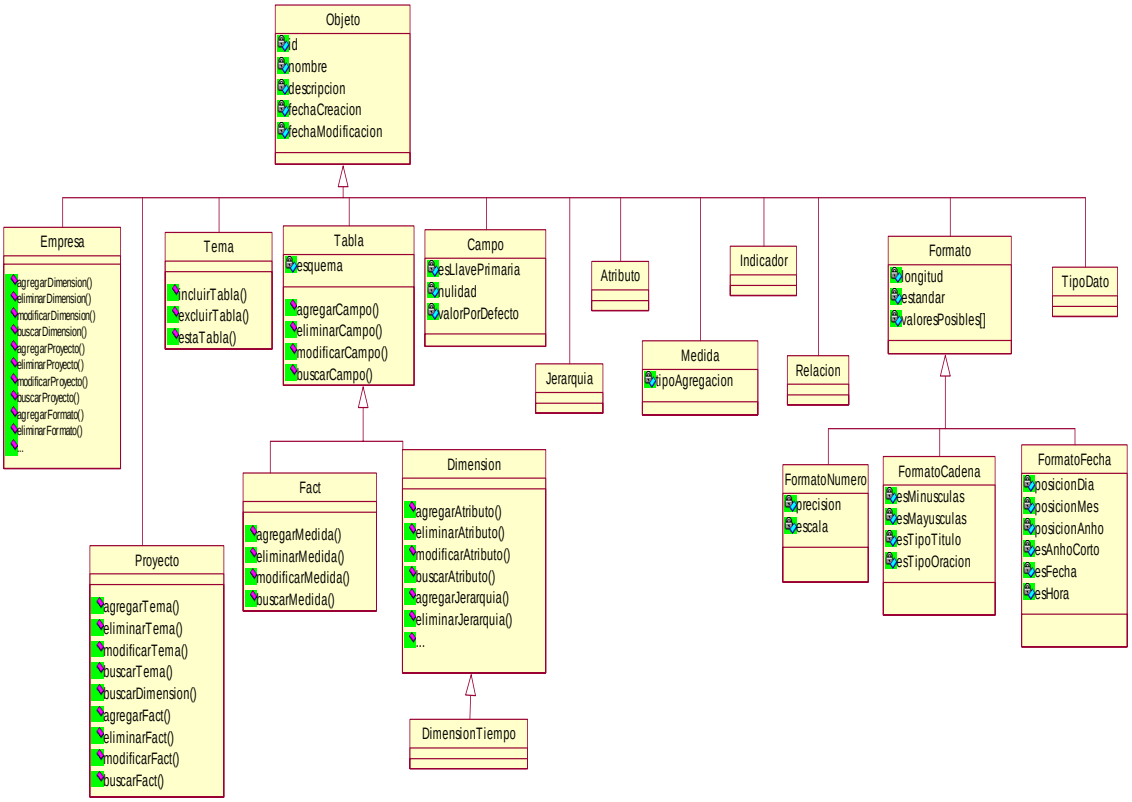


Figura 2. Diagrama de clases. Vista parcial de análisis.

3. tBUFEO, la infraestructura de pruebas de BUFEO

tBUFEO es la infraestructura de pruebas desarrollada expresamente para la plataforma de inteligencia de negocios BUFEO, con el objetivo de automatizar la ejecución de las pruebas de software. BUFEO está en su primera versión [7]; se trata de un producto comercial que evolucionará (crecerá y mejorará), por lo que resulta necesario y conveniente contar con una infraestructura de pruebas cuyo crecimiento se corresponda con la plataforma. El nombre de la infraestructura tiene una *t* al inicio para indicar que se trata de *testing* (pruebas); con *BUFEO* se indica que es exclusivo a BUFEO.

En la primera etapa del proyecto, las pruebas del software se realizaron de manera manual, a partir de un catálogo de pruebas registrado en una hoja electrónica. Para la segunda etapa se decidió automatizar la ejecución de las pruebas, desarrollándose el proyecto de la infraestructura de forma paralela al desarrollo de la plataforma. Esta decisión se basó en el hecho de que algunas pruebas se tornaron repetitivas —introducir los mismos datos para apreciar idénticos resultados—, porque eran sensibles a la automatización y porque, en algunos cambios de la plataforma, había que volver a probar varios aspectos del producto (pruebas de regresión) y pruebas de carga, tal como las entienden Pezzé [8] y Culbertson [9]. La infraestructura de pruebas, dentro de lo planteado, permitirá probar la plataforma BUFEO incluso a nivel de los modelos de negocios que más adelante se tiene previsto desarrollar sobre la plataforma.

En la construcción de tBUFEO se han incorporado, en primer lugar, las pruebas de caja negra (funcionales) para comprobar que se cumple los requisitos del producto; en segundo lugar, las pruebas referidas con grandes cantidades de datos (stress, volumen, carga) y, por último, un grupo de pruebas referentes a modelos de negocios que se desarrollarán sobre la plataforma (estas últimas pruebas están pendientes, porque el soporte a los modelos aún no ha sido desarrollado). Considerando la arquitectura por capas de BUFEO (interfaz, negocio y persistencia), las pruebas en la infraestructura se han desarrollado principalmente para la capa de negocios, dejando a un lado la capa de interfaz gráfica de usuario.

La infraestructura tBUFEO se planteó a partir de los siguientes requisitos (se enumeran los más relevantes):

- tBUFEO debe ser una infraestructura de software que haga posible ejecutar las pruebas de la plataforma de inteligencia de negocios BUFEO y que permita crecer en número de casos de prueba.
- tBUFEO debe permitir realizar pruebas funcionales y de manejo de grandes cantidades de datos, utilizando una definición de casos de prueba en archivos basados en XML.
- tBUFEO ha de posibilitar administrar la ejecución de los casos de prueba que se aplicarán sobre la plataforma.

A partir de aquí, se planteó un conjunto de casos de uso de tBUFEO teniendo como actor a un informático (probador), tal como se aprecia en la Figura 3. La arquitectura de la infraestructura se basa en un esquema que corresponde a la implementación de conductores de pruebas (test-drivers), de manera paralela a la propia arquitectura de la plataforma (Pezzé [8], McGregor [10]). Los programas que son los test-drivers se escriben de manera correspondiente con cada clase y los datos, al ser usados en las pruebas, se obtienen de archivos de datos en XML.

La infraestructura (ver Figura 3) tiene tres servicios principales: (i) ejecutar los casos de prueba que se encuentren registrados en la infraestructura —lo cual incluye seleccionar los casos que se van a probar y los casos que no se van a probar—; (ii) incorporar nuevos casos de prueba a la plataforma, siguiendo un esquema de adición de archivos en formato XML donde se tengan los datos que se usarán; (iii) eliminar casos de prueba que hayan quedado obsoletos a través de los sucesivos cambios de la plataforma. Este esquema se ha seguido para los tres grandes componentes de la plataforma BUFEO. Los casos de pruebas a nivel de unidad van desde probar algunos métodos complejos hasta probar grupos (cluster) de clases, e incluso componentes. A modo de ejemplo (ver Figura 4) se presenta la clase puEmpresa01, que se corresponde con una clase que probará a las clases asociadas a la clase Empresa, como son las clases ControlEmpresa y GestorEmpresa. Igualmente, se presenta la clase puJob01 (ver Figura 5), que corresponde a la clase que probará a las clases del BEJob, como son las clases Empresa, Proyecto, BEControlJob, ControlTema, ControlProyecto y ControlEmpresa; ambas clases de pruebas tienen como ámbito de prueba

un cluster de clases. Las Figuras 4 y 5 corresponden a los módulos de extracción y explotación. En el apéndice se presenta un extracto de dos clases que implementan test-drivers para probar empresas.

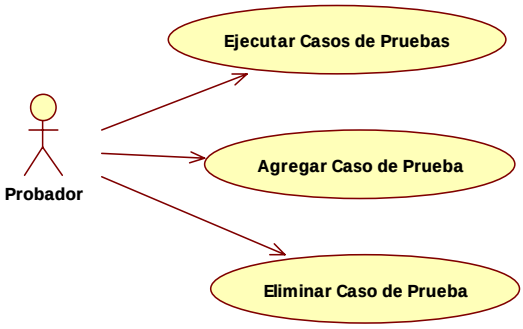


Figura 3. Diagrama de casos de uso de la infraestructura de tBufeo.

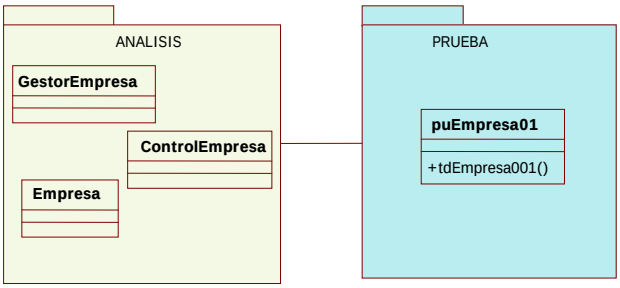


Figura 4. Test-driver de la clase Empresa.

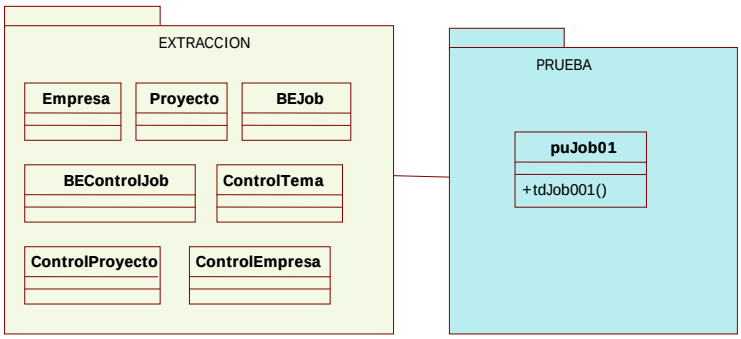


Figura 5. Test-driver de la clase Job.

4. Situaciones encontradas y lecciones aprendidas

Durante la implementación progresiva de tBUFEO se han realizado las pruebas (ejecución de tBUFEO) de la plataforma de inteligencia de negocios BUFEO con los resultados que siguen.

4.1. Situaciones encontradas

- Las funcionalidades del software que implican registro de datos a través de pantallas tipo formularios (o que implican un registro simple de datos) son las primeras que se han implementado. Para estos casos, el desarrollo de los test-drivers ha sido relativamente fácil y rápido, pues los desarrolladores de BUFEO han seguido los conceptos de tres capas a través del patrón Modelo Vista Controlador. Los test-drivers de la infraestructura implementan clases para manipular las clases Controladoras o Gestoras de acuerdo con cada situación.
- Las funcionalidades del software que implican el uso de una interfaz gráfica más elaborada, como suelen ser los servicios de modelado (por ejemplo, diagrama estrella o diagramas de transformación de datos), han representado una mayor complejidad que el caso anterior. Esta complejidad se debe al hecho de que la implementación de estos diagramas manipulan directamente la generación de los datos (almacenados en archivos en formato XML), dejando de lado la parte de las clases Controladoras o Gestoras.
- Las funcionalidades que implican la fijación de parámetros (criterios) para consultas (o reportes) son relativamente similares al primer caso. Sin embargo, la complejidad viene del lado de la comprobación de los resultados. En algunos casos se ha manejado una comprobación de resultados finales y en otros se ha hecho la comprobación de los datos que aparecen en la consulta (reporte). La automatización es posible, pero se ha dejado para una siguiente etapa la comprobación de los datos.
- En las pocas ocasiones en las que se ha tenido documentación no actualizada, inconsistente o inexistente, pues se trata de dos proyectos (la plataforma y la infraestructura) con relativa independencia y en paralelo, ha habido mayor trabajo, pues ha sido necesario revisar y entender de manera constante el código fuente para implementar los test-drivers.

4.2. Lecciones aprendidas

- Un programa bien desarrollado, que respete el diseño y que se encuentre debidamente documentado, permite trabajar con los test-drivers con mayor facilidad, rapidez y agrado; en caso contrario surge dificultad, lentitud y molestia. Ambas situaciones se encontraron en la plataforma. El equipo supervisó el proyecto para asegurarse de que el programa estuviese adecuadamente desarrollado en todos los casos. En los casos sencillos, los desarrolladores sí respetaron buenas prácticas, pero en algunos casos más complejos no todos lo hicieron.
- Para desarrollar una infraestructura de pruebas es necesario que el equipo que la construye tenga competencias técnicas en pruebas de software y, al menos, las mismas competencias que el equipo que desarrolló la plataforma (BUFEO). Esto es determinante para lograr construir test-drivers que interactúen con las clases que se utilizarán para manipular la porción de la plataforma que interesa, manteniendo la independencia entre ambos productos (plataforma e infraestructura de pruebas).
- Un programa con documentación deficiente supone que se tiene que revisar el código fuente y acarrea pruebas en falso, es decir, intentos de probar y encontrar que el test-driver no ha sido implementado de manera adecuada. Todo esto se traduce en demoras de trabajo y molestias para las personas involucradas en la construcción de la infraestructura.
- Los test-drivers que corresponden a casos con persistencia deben diseñarse de tal manera que sea posible volver a ejecutarlos y permitan hacer la comparación de manera automática. Por eso los test-drivers se han desarrollado, en primer lugar, fijando los datos en la persistencia y luego, realizando la operación con persistencia, verificando los valores finales con los valores esperados y, finalmente, borrando la operación.

5. Discusión final y trabajo futuro

Construir la infraestructura ha permitido identificar algunas lecciones sobre cómo se debe desarrollar y qué cosas se deben cuidar en un desarrollo de este tipo de producto, que

evolucionará en el tiempo. Para algunos casos ha sido posible trabajar directamente con la capa de Negocio, pero en otros casos no ha sido tan fácil porque no se han seguido de manera disciplinada estas separaciones en algunas porciones de BUFEO.

Una infraestructura de pruebas basada en test-drivers es útil para un software relativamente grande como es la plataforma de inteligencia de negocios. Más adelante se espera hacer uso de jUnit (http://www.junit.org/) y comparar el esfuerzo sin él. De manera análoga, para el caso de pruebas de volumen de datos se utilizará un software comercial y se comparará frente a lo desarrollado. En ambos casos se contará con equipos diferentes al que construyó tBUFEO.

Agradecimientos

Este proyecto ha sido parcialmente apoyado por PROCOM-CONCYTEC R.P. N° 080-2006-CONCYTEC-P, el Centro de Innovación y Desarrollo Emprendedor de la Pontificia Universidad Católica del Perú, y la empresa ACKLIS SAC.

Referencias

[1] IEEE, *IEEE Std. 610-12:1990. Glossary of Software Engineering Terminology*, IEEE, 1990.

[2] IEEE, *Guide to the Software Engineering Body of Knowledge*, IEEE, 2004.

[3] Presmman, R., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 2005.

[4] Sommerville I., *Software Engineering*, Addison Wesley, 2000.

[5] Futrell, S., Shafer, D. y Shafer, L., *Quality Software Project Management*, Prentice Hall, 2002.

[6] Valencia, G., *Subcontratación de las pruebas de software*, GreenSQA. http://www.greensqa.com/index.php?option=com_content&task=view&id=13&Itemid=30 (consultado en junio de 2008)

[7] González, D., Dávila, A., Basurto, C., Flores, L. y Morales, L., *Proyecto Sistema Software para la Aplicación de Inteligencia de Negocios en Organizaciones Medianas y Pequeñas*. http://www.acklis.com/bufeo/ (consultado en enero de 2008)

[8] Pezzé, M. y Young M., *Software Testing and Analysis, Process, Principles and Techniques*, Wiley, 2008.

[9] Culbertson, R., Brown C. y Cobb G., *Rapid Testing*, Prentice Hall, 2002.

[10] McGregor, J., *CMU/SEI-2001-TR-02. Testing a Software Product Line*, Software Engineering Institute. Carnegie-Mellon University, 2001.

Apéndice A

Programa ejemplo de los test-drivers de la infraestructura.

```
/* METODO: tdEmpresa001
 * OBJETIVO: Crear la Empresa.
 * @return boolean, true si la prueba fue correcta, caso contrario es false. */
public boolean tdEmpresa001() {
 boolean blnResultadoObtenido = true;
 long lngIdEmpresa = 0;
 try {
  /*1.- Creando Empresa */
  Empresa objEmpresa1 = cargardatosEmpresa.cargarDatosEmpresa001("XEMP");
  lngIdEmpresa = objEmpresa1.getId();
  /*2.- Obteniendo datos de Empresa creada*/
  GestorEmpresa objGestorEmpresa = new GestorEmpresa();
  Empresa objEmpresa2 = objGestorEmpresa.buscarEmpresa(objEmpresa1.getId());
  /*3.- Comprobando*/
  if((objEmpresa1.getId()==objEmpresa2.getId())&&
  (objEmpresa1.getNombre().equals(objEmpresa2.getNombre()))&&
  (objEmpresa1.getDescripcion().equals(objEmpresa2.getDescripcion()))&&
  (objEmpresa1.getDireccion().equals(objEmpresa2.getDireccion()))&&
  (objEmpresa1.getRUC().equals(objEmpresa2.getRUC()))&&
  (objEmpresa1.getRubro()== objEmpresa2.getRubro())&&
  (objEmpresa1.getRepresentanteLegal().equals(objEmpresa2.getRepresentanteLegal())))
  { blnResultadoObtenido = true;
  } else {
  blnResultadoObtenido = false; }
  /*4.- Eliminando datos cargados*/
  eliminardatosEmpresa.eliminarDatosEmpresa(objEmpresa1);
 } catch (Exception e) {
  e.printStackTrace(); }
```

```
 return blnResultadoObtenido;
}
/**
 * METODO:  cargarDatosEmpresa001
 * OBJETIVO: Crear la Empresa.
 * @param astrIdentificadorEmpresaxPrueba.
 * @return Empresa, devuelve la Empresa creada. */
 public static Empresa cargarDatosEmpresa001 ( String astrIdentificadorEmpresaxPrueba )
throws Exception{
 AppletServlet objAppletServlet = new AppletServlet();
 /*1.- Creando Empresa*/
 Empresa objDatosEmpresa = null;
 long lngIdEmpresa = 0;
 try {
  /*2.- Obtiene Empresa*/
objDatosEmpresa=ControlEmpresa.agregarEmpresa("BUFEO"+strIdentificadorEmpresaxP
rueba, "descripcion", "dir","111", 3456, "www.bipucp.pucp.edu.pe", "su abogado",1);
  lngIdEmpresa = objDatosEmpresa.getId();
} catch (Exception e) {
  e.printStackTrace();
}
/*3.- Devolviendo la empresa creada*/
  return objDatosEmpresa;
}
```

Perfiles del ciclo de vida del software para pequeñas empresas: los informes técnicos ISO/IEC 29110

José A. Calvo-Manzano, Javier Garzás, Mario Piattini,
Francisco J. Pino, Jesús Salillas, José Luis Sánchez
Grupo de Trabajo 24 del AENOR 71/SC7
jacalvo@fi.upm.es; Javier.Garzas@kybeleconsulting.com; Mario.Piattini@uclm.es;
FcoJose.Pino@alu.uclm.es; Jesus.Salillas@esi.es; Jose.Luis.Sanchez@stl.es

Abstract

Small and medium enterprises are a very important piece in the world economy. The majority of software development is carried out by small —and medium— sized software development organisations. These organizations need efficient Software Engineering practices adapted to their sizes and business type. In this sense, and to support software process improvement in very small software enterprises, ISO through the SC7 has established a working group (WG24). The objective of this group is to create standard ISO profiles in order to apply them (software process improvement) in small software enterprises. This working group is establishing a common frame to describe software life cycle profiles to be used by very small enterprises.

Key words: software process improvement, small software enterprises, ISO/IEC 29110.

Resumen

Las pymes son una pieza clave en la economía mundial. La mayoría del software desarrollado lo realizan pymes. Este tipo de empresas necesita prácticas eficientes de ingeniería del software adaptadas a su tamaño y tipo de negocio. En este sentido, y para apoyar a las pequeñas empresas en sus esfuerzos de mejora de procesos, ISO ha conformado el grupo de trabajo denominado SC7-WG24 con el objetivo de que sus estándares para la mejora de procesos de software se puedan aplicar a pequeñas empresas desarrolladoras de software. Este grupo de trabajo está estableciendo un marco común para describir perfiles evaluables del ciclo de vida de software para su uso en las pymes.

Palabras clave: ciclo de vida del software, mejora de procesos de software, calidad de software, pymes, ISO/IEC 29110.

1. Introducción

Cada vez más, la industria del software representa una actividad económica de suma importancia en la mayoría de los países del mundo. En Europa representa el 8% del PIB y el 6% de los puestos de trabajo; en España esta importancia está creciendo más aún, por la actual tendencia que se está experimentando hacia modelos de negocio basados en fábricas

de software y *nearshoring* [7, 14]. A nivel mundial, esta industria está formada por micros, pequeñas y medianas empresas desarrolladoras de software –pymes– que suponen cerca del 90% de los negocios formales y que generan entre el 40% y el 50% del empleo total. Estudios como el desarrollado en [15] muestran que la mejora de procesos de software (SPI) es una actividad que las pequeñas empresas desean implementar para incrementar la calidad y capacidad de sus procesos y, en consecuencia, la calidad de sus productos y servicios. Este mismo estudio evidencia que, para mejorar sus procesos, las pequeñas empresas están utilizando estándares de procesos de organizaciones como el Software Engineering Institute (SEI) y la International Organization for Standardization (ISO), entre los que destacan CMMI-DEV [19], ISO 12207 [6], ISO 15504 [5] e ISO 9001 [4]. Sin embargo, diversos estudios [3, 17, 20] apuntan a que la aplicación de estos estándares en las pymes es difícil, ya que supone para ellas una gran inversión en dinero, tiempo y recursos, dado que estos modelos están orientados a las grandes organizaciones. Además, las recomendaciones de estos modelos son complejas de aplicar y el retorno de la inversión se da a largo plazo. Asimismo, en las pequeñas empresas la aplicación de estos modelos se agrava aún más, ya que existe un problema "cultural" importante cuando se quiere "importar" y adoptar, sin más, modelos creados para otro tipo de organizaciones; como señala [23], si el proceso no "casa" con la cultura de la organización, será rechazado por el "cuerpo" organizativo, como sucede en los trasplantes de órganos. Un problema parecido se expone en la investigación de [1], en la que se destaca la importancia de las diferencias culturales en el éxito de la mejora de procesos de software entre las empresas de EEUU y Europa.

Además, según [2], este tipo de empresas necesita, para la construcción de sus productos, prácticas eficientes de ingeniería del software adaptadas a su tamaño y tipo de negocio. En este sentido, y para apoyar a las pequeñas empresas en sus esfuerzos de mejora de procesos, ISO ha constituido el grupo de trabajo SC7-WG24, cuyo objetivo es que sus estándares de procesos de software (o adaptaciones de estos) se puedan aplicar a pequeñas empresas desarrolladoras de software. Este grupo está estableciendo un marco común para describir perfiles evaluables del ciclo de vida de software para su uso en Very Small Enterprises (VSE, una organización de menos de 25 empleados). Estos perfiles se publicarán en el año 2010 con el nombre ISO/IEC 29110. Un perfil es un conjunto de

procesos para ayudar a aplicar una norma ISO; desde un punto de vista práctico, un perfil es una especie de lista de material compuesto de partes de normas como ISO/IEC 12207 o ISO/IEC 15504 [8].

En este artículo los miembros del grupo de trabajo 24 del AENOR 71/SC7 presentan una descripción general de los actuales informes técnicos relativos a ISO/IEC 29110. En el epígrafe segundo aparecen otras propuestas para la mejora de procesos en pequeñas empresas. En el tercero se describe el grupo de trabajo WG 24 y el cuarto epígrafe muestra la visión general de los informes técnicos ISO/IEC 29110. Finalmente, se presentan las conclusiones y futuros trabajos.

2. Propuestas para la mejora de procesos en pymes

A continuación, y de acuerdo con el reciente estudio presentado en [7], se enumeran las principales iniciativas de estándares y propuestas regionales relacionadas con la mejora de procesos de software en pequeñas empresas:

- En la Unión Europea se han impulsado iniciativas como ESSI (European Software and System Initiative), que ha promovido diferentes proyectos para fortalecer SPI en pequeñas empresas, como SPIRE [2] y TOPS[3].
- En México se creó el Modelo de Procesos para la Industria del Software (MoProSoft) [12], basado en ISO 12207, CMM e ISO 9001, y el método de evaluación de procesos para la industria de software (EvalProSoft) [11], basado en ISO 15504.
- En Brasil se desarrolló el proyecto MPS-BR [22]. Su modelo de referencia MR-MPS está basado en ISO 12207 y CMMI; su método de evaluación MA-MPS está basado en ISO 15504.
- Para Iberoamérica, actualmente se está desarrollando el proyecto COMPETISOFT [13]. Su modelo de referencia está basado en ISO 12207, CMM, ISO 9001, MANTEMA [16] y Métrica V3 [9]. Como método de evaluación, sugiere cualquiera que sea conforme a ISO 15504 y el modelo de gestión de mejora está influido por IDEAL [10] y SCRUM [21].

[2] http://www.cse.dcu.ie/spire/
[3] http://www.cordis.lu/esprit/src/27977.htm

- También el SEI ha creado un Consorcio Internacional de Investigación de Procesos (IPRC) con el fin de mejorar procesos para Small Settings (IPSS) [18]. Small Settings hace referencia a proyectos pequeños (menos de 20 personas), pequeñas organizaciones (menos de 50 personas) y/o pequeñas empresas (menos de 100 personas).
- El ESI ha diseñado el modelo I.T.Mark, que acredita la calidad y madurez de los procesos de las pequeñas y las microempresas de T.I. Se basa en modelos internacionalmente reconocidos, como el CMMI e ISO-17799:2005, y se ha aplicado con éxito en Europa, Asia e Iberoamérica.

Estos trabajos pretenden abordar la mejora de procesos en pequeñas empresas, teniendo presentes los estándares de ISO y los modelos del SEI. Así pues, el trabajo desarrollado por el SC7-WG24 es una contribución muy importante en el esfuerzo por hacer que los actuales estándares de ingeniería de software de ISO sean más accesibles a pequeñas empresas, ya que, según el estudio presentado en [8], a las pequeñas empresas les resulta difícil relacionar las normas ISO con sus necesidades de negocio y, por tanto, justificar la aplicación de estas normas. Además, la mayor parte de estas empresas no puede soportar los recursos necesarios para tal aplicación, en términos de personal, coste y tiempo, ni ver el beneficio neto que pueden obtener cuando establecen los procesos del ciclo de vida del software.

3. El grupo de trabajo WG 24

ISO nació en 1947 para facilitar la coordinación internacional de las normas técnicas en los diferentes campos de la industria. Pueden ser miembros de ISO todos aquellos países del mundo que lo deseen, representados a través de su organismo nacional de normalización, por ejemplo: ANSI (American National Standards Institute) por EEUU o AENOR (Asociación Española de Normalización y Certificación) por España. Los trabajos de elaboración de normas están encomendados a los Comités Técnicos (TC), que suelen subdividirse en Subcomités (SC) y estos, a su vez, en Grupos de Trabajo (WG) para desarrollar temas específicos (Figura 1).

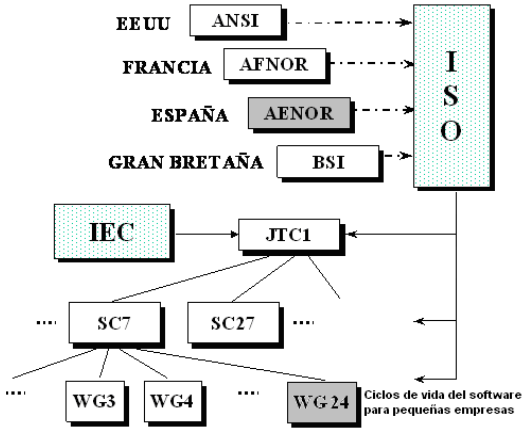


Figura 1. Grupos de trabajo (WG) y estructura de ISO.

En algunas áreas, ISO colabora con otras organizaciones; así, en el campo de las tecnologías de la información, forma, junto con la International Electrotechnical Commission (IEC) el Joint Technical Committee 1 (JTC1), que se divide en varios subcomités, entre ellos, el SC7 de Ingeniería del Software y Sistemas, que posee diferentes grupos de trabajo (WG), como el WG 24. El grupo WG 24 nació en la sesión plenaria del SC7 en Finlandia en el año 2005 con el propósito de gestionar y desarrollar el trabajo para alcanzar los siguientes objetivos [8]:

- Hacer los estándares de ingeniería de software actuales más accesibles a las pequeñas empresas.
- Proporcionar documentación que requiera un mínimo esfuerzo de adaptación.
- Proveer documentación armonizada integrando estándares disponibles, tales como estándares de proceso, evaluación, calidad y modelado, y también productos de trabajo, entregables y herramientas.
- Tener en cuenta, si es preciso, las nociones de niveles de madurez y capacidad presentadas en ISO/IEC 15504 y CMMI.

A partir de los trabajos del grupo WG 24, orientados a satisfacer estos objetivos, se ha desarrollado un conjunto de informes técnicos preliminares que son la base sobre la cual se estructura el futuro estándar ISO/IEC 29110; de ellos se ofrece una perspectiva general en las siguientes secciones.

4. Visión general de los informes técnicos ISO/IEC 29110

Los informes técnicos ISO/IEC 29110 se denominan "Ingeniería de Software. Perfiles de Ciclo de Vida para Empresas Muy Pequeñas" (Software Engineering. Lifecycle Profiles for Very Small Enterprises, VSE). Un informe técnico es un documento publicado por ISO/IEC que ayuda a la comprensión y al uso de la parte normativa de un estándar. En el ámbito de ISO/IEC 29110, los informes técnicos se utilizan para presentar las guías sobre la implementación de perfiles en VSE. Estas guías proporcionan información práctica para facilitar la implementación de los perfiles definidos. Para ISO/IEC 29110, las guías (directrices) serán promulgadas como informes técnicos.

Un perfil es un subconjunto de uno o más estándares necesarios para llevar a cabo una función particular, por ejemplo, identificar un ciclo de vida del software adaptado y adecuado a las necesidades del negocio de una VSE. Un perfil incluye típicamente elementos extraídos de los estándares y diseñados para proporcionar una implementación coherente de funcionalidades específicas. Los perfiles están diseñados para proporcionar:

- Conceptos de diferentes estándares.
- Un sistema de referencia que es significativo para clientes, usuarios y proveedores.
- Una base para el desarrollo de la evaluación de conformidad de forma objetiva, uniforme y reconocida internacionalmente.

Un Perfil Internacional Estandarizado (International Standardized Profile, ISP) es un documento acordado internacionalmente que incluye las especificaciones de uno o más perfiles.

La conformidad con los perfiles es la forma mediante la que las VSE muestran y documentan el uso y la comprensión de los estándares internacionales.

Los requisitos clave que el WG 24 planea satisfacer con este nuevo estándar son:

- Proporcionar a las VSE un reconocimiento como productoras de sistemas de software de calidad con unos costes muy reducidos en cuanto a la implementación y el mantenimiento de toda una colección de estándares de ingeniería de software y sistemas, o de la realización de evaluaciones exhaustivas.
- Elaborar guías que sean fáciles de entender, asequibles y utilizables por las VSE.
- Producir un conjunto de perfiles, los cuales construyan o mejoren procesos existentes de la VSE, u ofrezcan orientación en el establecimiento de esos procesos.

- Atender las necesidades del mercado de las VSE, permitiendo perfiles y niveles para dominios específicos.
- Dar ejemplos con el fin de fomentar que las VSE adopten y sigan los procesos que conducen a un software de calidad, y que consideren las necesidades, los problemas y los riesgos de sus dominios.
- Proporcionar una línea de base de cómo múltiples VSE pueden trabajar de forma conjunta o ser evaluadas como un equipo de proyecto, sobre proyectos que pueden ser más complejos de los que cualquier VSE puede realizar individualmente.
- Elaborar perfiles y guías escalables que están en conformidad con ISO/IEC 12207, 15504 y/o ISO 9001:2000, y permitir que la evaluación sea posible con un mínimo de rediseño de los procesos de la VSE.

La Figura 2 muestra la estructura de ISO/IEC 29110.

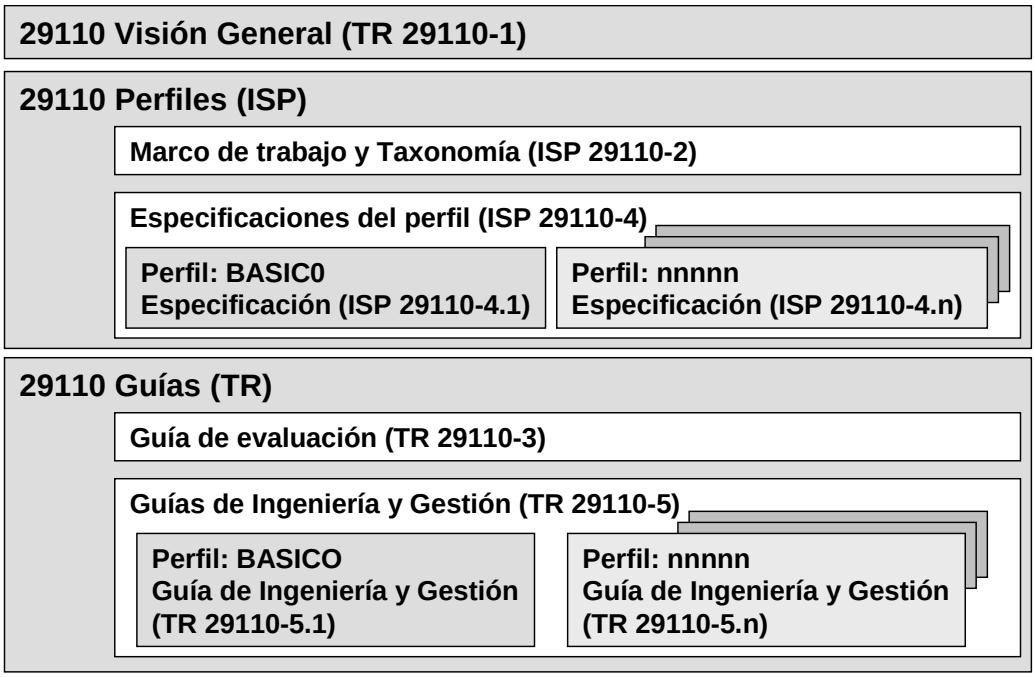


Figura 2. Familia de documentos ISO/IEC 29110.

4.1. Visión general (TR 29110-1)

La visión general ofrece los conceptos principales necesarios para comprender y utilizar los documentos de ISO/IEC 29110. Introduce los aspectos de negocio, características y requisitos de VSE, y aclara la razón de ser de los perfiles específicos, documentos,

estándares y guías de VSE. Asimismo, presenta los conceptos de proceso básico, ciclo de vida y estandarización, y la familia de documentos ISO/IEC 29110.

4.2. Perfiles (ISP)

Los perfiles se definen con el objetivo de empaquetar referencias a y/o partes de otros documentos de manera formal para adaptarlos a las necesidades y características de las VSE. Preparar perfiles es un proceso definido de ISO/IEC JTC1, lo cual implica producir dos tipos de documentos:

- Marco de trabajo y taxonomía (TR29110-2). Este documento establece la lógica detrás de la definición y aplicación de los perfiles. Especifica los elementos comunes a todos los perfiles (estructura, conformidad, evaluación) e introduce la taxonomía (catálogo) de los perfiles ISO/IEC 29110. Este documento está dirigido a autores y revisores de ISP (*International Standardized Profiles*), autores de otras partes y de otros perfiles orientados a VSE. El marco de trabajo y la taxonomía se pueden aplicar a los perfiles identificados según TR 29110-2.
- Especificaciones de perfil (TR29110-4). Por cada perfil hay un documento de especificación de perfil, identificado como 29110-4.X, donde la X es el número asignado al perfil. Su propósito es proporcionar la composición definitiva de un perfil, proporcionar enlaces normativos al subconjunto normativo de estándares (por ejemplo, ISO 12207) usados en el perfil y proporcionar enlaces informativos (referencias) a documentos de "entrada" (por ejemplo, 90003, SWEBOK, de herramientas y otro material de apoyo.

El documento 29110-4.1 "Especificación. Perfil Básico PMI" está dirigido a autores/proveedores de guías. Es un ejemplo de una especificación de perfil. Su propósito es definir una guía de gestión de proyectos y desarrollo de software para un subconjunto de procesos de ISO/IEC 12207, apropiada para las características y necesidades de una VSE. La principal razón para incluir la gestión de proyectos es que el corazón del negocio de la VSE es el desarrollo de software, y su éxito financiero depende de los beneficios del proyecto.

4.3. Guías

Las guías contienen directrices de aplicación (de dominio específico) sobre cómo realizar los procesos para alcanzar los niveles de madurez (por ejemplo, actividades recomendadas,

medidas, técnicas, plantillas, modelos y métodos, entre otros). Las guías se desarrollan para la implementación del proceso y para la evaluación basada en aspectos de dominio, prácticas y riesgos del negocio. Las guías están dirigidas a VSE y deberían ser accesibles por la VSE en términos de estilo y costo. Hay dos tipos de guías:

- Guía de evaluación (TR29110-3), que describe el proceso que se ha de seguir para realizar una evaluación que determine las capacidades de proceso y la madurez organizativa. Es decir, sirve cuando una organización desea una ejecución de evaluación con el fin de obtener un perfil de capacidad de los procesos implementados y/o un nivel de madurez organizativa. También es aplicable a una situación en la que el cliente solicita a un tercero la ejecución de la evaluación para obtener un perfil del nivel de capacidad del proceso implementado por el proveedor del desarrollo y mantenimiento del software. Igualmente, es adecuada para la autoevaluación y es aplicable a todos los perfiles identificados conforme a TR 29110-3. La metodología de la evaluación es consistente y acorde con los principios establecidos en ISO/IEC 15504-2.
- Guía de ingeniería y gestión (TR29110-5), que proporciona orientación sobre su implementación y uso o sobre un perfil. Para cada perfil, existe un documento de guía de ingeniería y gestión identificado como 29110-5.x, siendo x el número asignado al perfil (este número coincide con el número que se asigna a la especificación del perfil). Está dirigido a la VSE (personal directivo y técnico), a las organizaciones relacionadas con la VSE (centros de transferencia de tecnología, ministerios de industria del gobierno, estándares nacionales, consorcios y asociaciones, uso académico para la capacitación, autores de productos de software liberados, programas de computación destinados a uso educacional, el comprador y los proveedores, entre otros). Se debe destacar que, por ir dirigido a la VSE, se elabora en un lenguaje más sencillo y de más fácil comprensión que el que se utiliza habitualmente en los estándares.

El documento 29110-5.1 “Guía de ingeniería y gestión. Perfil básico” es un ejemplo de guía de ingeniería y gestión. Este perfil se compone de dos procesos: Gestión de Proyectos e Implementación de Software. El cliente proporciona una Descripción del Producto como una entrada y recibe una Configuración de Software como un resultado de

la ejecución del proceso de Implementación de Software, el cual es controlado por el proceso de Gestión de Proyectos (Figura 3).

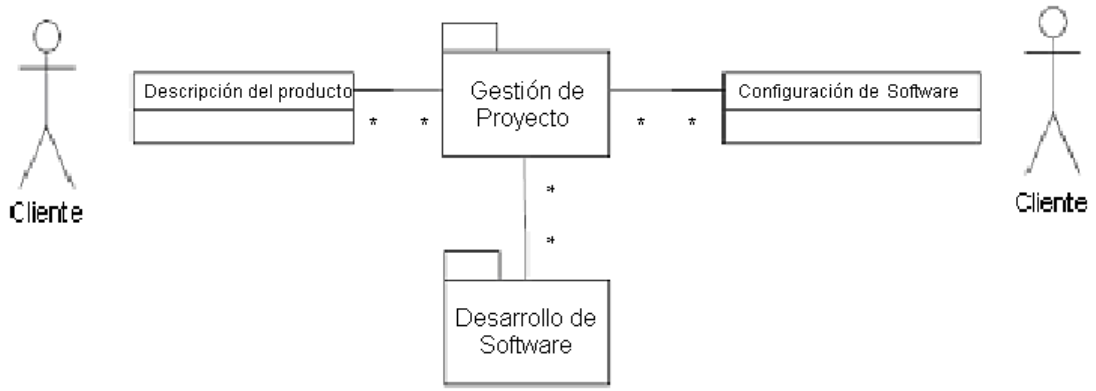


Figura 3. Estructura de la guía del perfil básico.

El proceso de Gestión de Proyectos recibe la Descripción del Producto para elaborar el Plan del Proyecto que sirve para orientar la ejecución del proceso de Implementación del Software. El subproceso de Evaluación y Control de Proyectos compara el Registro del Estado del Progreso del proyecto frente al Plan del Proyecto para eliminar las desviaciones a través de Acciones Correctivas o cambios al Plan del Proyecto. Por último, el subproceso del Cierre del Proyecto entrega la Configuración del Software y obtiene el Documento de Aceptación del cliente para formalizar el cierre del proyecto.

El proceso de Implementación de Software comienza con el subproceso de Iniciación, que recibe el Plan del Proyecto que guía la ejecución de los subprocesos Análisis de Requisitos de Software, Diseño de Arquitectura de Software y Diseño Detallado, Construcción de Software, Integración y Pruebas de Software, y Entrega del Producto. Las tareas de verificación, validación y prueba se incluyen en el trabajo de los subprocesos para eliminar defectos de los productos. Se establece un Repositorio de Proyecto que protege los productos de trabajo y controla sus versiones y estado.

5. Conclusiones y trabajo futuro

Este artículo ha presentado el trabajo que está llevando a cabo ISO para elaborar un estándar internacional de mejora de procesos de software especialmente adaptado a las necesidades de las pymes, facilitando que puedan abordar procesos de mejora de una forma más adecuada a sus estructuras organizativas y a sus negocios. Esta mejora, en cuanto a

calidad y productividad, es fundamental para competir en mejores condiciones y con mejores resultados en un entorno cambiante, resultado de la globalización económica. Como se ha expuesto, existen numerosas barreras que hacen que la aplicación de los modelos más tradicionales de mejora de procesos (CMMI, ISO 15504, etc.) resulte costosa en términos económicos y de esfuerzo en el contexto de las pequeñas empresas, lo cual dificulta su adopción y difusión en las mismas

Como se desprende del artículo, las perspectivas que se presentan son atractivas. Animamos a todas las pymes interesadas en esta iniciativa para que contacten con el grupo de trabajo GT24 de AENOR y contribuyan, así, a que las características de las pymes españolas se tengan aún más en cuenta a la hora de definir las guías y perfiles.

Referencias

[1] Dyba, T., "An Empirical Investigation of the Key Factors for Success in Software Process Improvement", *IEEE Transactions on Software Engineering*, vol. 31, nº 5, pp. 410-424, 2005.

[2] Fayad, M.E., Laitinen, M.y Ward, R. P., "Software Engineering in the Small", *Communications of the ACM*, vol. 43, nº 3, pp. 115-118, 2000.

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

[4] ISO, *ISO 9001:2000. Quality Management Systems. Requirements*, International Organization for Standardization, 2000.

[5] ISO, *ISO/IEC 15504-2:2003/Cor.1:2004(E). Information Technology. Process Assessment. Part 2: Performing an Assessment*, International Organization for Standardization, 2004.

[6] ISO, *ISO/IEC FDIS 12207:2007(E). Systems and Software Engineering. Software Life Cycle Processes*, International Organization for Standardization, 2007.

[7] ITECO, *Estudio sobre la certificación de la calidad como medio para impulsar la industria de desarrollo del software en España*, Instituto Nacional de Tecnologías de la Comunicación, 2008.

[8] Laporte, C., Alexandre, S. y Renault, A., "Developing International Standards for VSEs", *IEEE Computer*, vol. 41, nº 3, pp. 98-101, 2008.

[9] MAP, *Métrica Versión 3. Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información,* Ministerio de Administraciones Públicas, 2007.
[10] McFeeley, R., *IDEAL: A Users Guide for Software Process Improvement, Handbook CMU/SEI-96-HB-001,* Software Engineering Institute. Carnegie Mellon University, 1996.
[11] Oktaba, H., *Método de evaluación de procesos para la industria de software. EvalProSoft - Versión 1.1, marzo de 2004. NMX-I-006/(01 al 04)-NYCE-2004,* Organismo nacional de normalización y evaluación de la conformidad. NYCE, 2004.
[12] Oktaba, H., "MoProSoft®: A Software Process Model for Small Enterprises", *Proceedings of the First International Research Workshop for Process Improvement in Small Settings. Pittsburgh*, pp. 93-101, 2006.
[13] Oktaba, H., García, F., Piattini, M., Pino, F., Alquicira, C. y Ruiz, F., "Software Process Improvement: the COMPETISOFT Project", *IEEE Computer,* vol. 40, nº 10, pp. 21-28, 2007.
[14] Piattini, M. y Garzás, J., *Fábricas de software: experiencias, tecnologías y organización,* Ra-Ma, 2007.
[15] Pino, F., García, F. y Piattini, M., "Revisión sistemática de mejora de procesos de software en micro, pequeñas y medianas empresas", *Revista Española de Innovación, Calidad e Ingeniería del Software,* vol. 2, nº 1, pp. 6-23, 2006.
[16] Polo, M., Piattini, M., Ruiz, F. y Calero, C., "MANTEMA: A Software Maintenance Methodology Based on the ISO/IEC 12207 Standard", *Proceedings of the 4th IEEE International Symposium and Forum on Software Engineering Standards. Curitiba, Brasil,* pp. 76-81, 1999.
[17] Saiedian, H. y Carr, N., "Characterizing a Software Process Maturity Model for Small Organizations", *ACM SIGICE Bulletin,* vol. 23, nº 1, pp. 2-11, 1997.
[18] SEI, *Applying CMMI® in Small Settings (ACSS Project),* Software Engineering Institute, 2005.
[19] SEI, *CMMI for Development. Version 1.2. Technical Report CMU/SEI-2006-TR-008,* Software Engineering Institute, 2006.
[20] Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P. y 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.

[21] Takeuchi, H. y Nonaka, I., "The New Product Development Game", *Harvard Business Review*, nº 1, pp. 1-12, 1986.

[22] Weber, K., Araújo, E., Rocha, A., Machado, D. Scalet y Salviano, C., "Brazilian Software Process Reference Model and Assessment Method", en *Computer and Information Sciences*, Springer Berlin/Heidelberg, pp. 402-411, 2005.

[23] Zahran, S., *Software Process Improvement: Practical Guidelines for Business Success*, Addison-Wesley, 1998.

Estudio experimental de la conversión entre las unidades de medición funcional del software puntos de casos de uso e IFPUG

Juan J. Cuadrado-Gallego, María J. Domínguez-Alda,
Marian Fernández de Sevilla, Miguel Ángel Lara
Departamento de Ciencias de la Computación. Universidad de Alcalá
{jjcg, mariajose.dominguez, marian.fernandez}@uah.es

Abstract

The objective of this research is to find a mathematical equation that allows convert software functional size measured with IFPUG unit into Use Cases Points unit. A database built explicitly for this research is used to obtain the equation. Both a lineal and no lineal mathematical equations have been founded, in order to obtain the most accuracy results. Another goal for this research has been to develop a process to obtain this kind of data in an academic environment.

Key words: software engineering, software measurement, IFPUG, Use Case Points.

Resumen

El objetivo de esta investigación es la búsqueda de una función matemática que permita convertir mediciones del tamaño funcional del software realizadas en unidades IFPUG a unidades puntos de casos de uso. Para ello se parte de un conjunto de datos obtenidos en esta investigación. La búsqueda de la función matemática se ha realizado tratando tanto de obtener una ecuación lineal como de buscar ecuaciones no lineales que puedan dar una solución más correcta y precisa al problema. Como parte de esta investigación también se ha abordado el problema de la obtención de datos para este tipo de estudios, proponiéndose un procedimiento repetible y contrastado para la obtención de datos fiables en un entorno académico.

Palabras clave: ingeniería del software, medición del software, IFPUG, puntos de casos de uso.

1. Introducción

La medición del tamaño del software que va a ser desarrollado en un proyecto es una de las magnitudes más importantes para la correcta gestión del mismo. Las primeras unidades que se definieron para medir el tamaño del software fueron las líneas de código fuente. Aunque se trata de una unidad útil cuando se utiliza para analizar diferentes aspectos, como el índice de errores o de productividad de un equipo, presenta el problema de que solo se

puede aplicar una vez que se ha construido el software. Por esta razón, la definición de una magnitud que permitiese medir el tamaño del software antes de construirlo se convirtió en una tarea esencial.

La respuesta a este problema la dio el investigador de IBM Alan Albrecht [1], que definió una nueva unidad de medida: el tamaño funcional del software, a la que denominó puntos de función. Esta unidad se aplica sobre los documentos que sí existen en las fases iniciales de un proyecto, como son la especificación de requisitos y los modelos de análisis, para obtener una medida de la cantidad de funcionalidad que el software entregará al usuario. Albrecht estableció —y posteriormente se comprobó de forma experimental [2]— que dicha medida estaba directamente relacionada con el número de líneas de código que tendría el software una vez realizado y, por lo tanto, que era una unidad correcta de medida del tamaño del software que, además, se podía obtener al comienzo del proyecto.

El método de casos de uso permite documentar los requisitos de un sistema en términos de actores y casos de uso, proporcionando uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico [3]. Debido a la importancia de esta técnica en el análisis orientado a objetos para capturar y describir los requerimientos funcionales de un sistema, surgió la necesidad de desarrollar un método de cálculo del tamaño funcional específicamente diseñado para esa técnica. Uno de los primeros resultados lo obtuvo en 1993 Gustav Karner [4], que desarrolló el método de puntos de casos de uso, como una extensión de los puntos de función IFPUG.

Este método cuantifica las características del sistema, tanto funcionales como no funcionales, a través de los siguientes pasos:

1. Categorización de los actores del modelo de casos de uso:

- Simple. Sistema externo que interactúa a través de una interfaz de programación definida y conocida (API) (Factor 1).
- Promedio. Sistema externo que interactúa a través de un protocolo (conjunto de reglas que especifican el intercambio de datos u órdenes durante la comunicación entre las entidades que forman parte de una red), como TCP/IP (Factor 2).
- Complejo. Usuario físico que interactúa a través de una interfaz gráfica de usuario (GUI) o sitio web (Factor 3).

Esta categorización da como resultado el número total de UAW (Unadjusted Actor Weight), contando el número de actores existentes en cada categoría, multiplicando cada resultado por el factor de ajuste correspondiente y realizando la suma de los resultados.

2. Categorización de los casos de uso:

- Simple. Tres transacciones o menos (Factor 5).
- Promedio. Entre cuatro y siete transacciones (Factor 10).
- Complejo. Más de siete transacciones (Factor 15).

Las transacciones son un grupo de actividades que se ejecutan de forma completa (éxito) o bien se vuelve al estado previo a la ejecución de la transacción (fracaso), quedando siempre el sistema en un estado consistente.

La categorización de los casos de uso permite conocer el valor del UUCW (Unadjusted Use Case Weights) de manera análoga a como se realizaba en el punto anterior, mediante el conteo del número de casos de uso en cada categoría, multiplicando estos por el factor de ajuste correspondiente y realizando la suma de los resultados.

Utilizando los resultados obtenidos en los dos pasos anteriores, se obtiene el número total de UUCP (Unadjusted Use Case Points) o puntos de casos de uso desajustados, mediante la fórmula:

$$UUCP = UAW + UUCW$$

3. Ajuste del valor UUCP

Mediante el uso de diversos factores técnicos y ambientales (no funcionales), a los cuales se asigna un valor comprendido entre cero y cinco, dependiendo de su influencia en el desarrollo del proyecto, se obtiene el valor TCF (Technical Complexity Factor) y EF (Environmental Factor), mediante las formulas:

$$TCF = 0{,}6 + (0{,}01 * TFactor)$$

$$EF = 1{,}4 + (-0{,}03 * EFactor)$$

TFactor y EFactor se calculan multiplicando el valor de cada uno de los factores de cada tabla por su peso y obteniendo la suma total.

Finalmente, mediante el uso de estos factores de ajuste TCF y EF, se obtiene el valor ajustado de puntos de casos de uso UCP (Use Case Points):

$$UCP = UUCP * TCF * EF$$

Uno de los problemas más importantes que actualmente queda por resolver en el campo de la medición del software es la conversión de unidades de medición funcional del software. El origen del problema reside en que existen muchas industrias en bastantes países que desde hace tiempo utilizan IFPUG como unidad de medida del software, habiendo desarrollado extensas bases de datos que les sirven para realizar correctamente las planificaciones de proyectos futuros. En la generación de esas bases de datos, para que los resultados obtenidos sean fiables, se necesita un gran número de años; por eso, la única solución realista que permitiría la introducción de otros métodos sería encontrar un factor de conversión de las medidas realizadas con IFPUG a puntos de casos de uso, de tal manera que se pudieran convertir automáticamente.

2. Obtención de los datos

2.1. Problemas en la obtención de los datos

Uno de los principales problemas de este tipo de investigaciones radica en la obtención de conjuntos de datos que, tanto en calidad como en tamaño, permitan hacer un análisis fiable y riguroso sobre ellos. El origen del problema se encuentra en el coste de la obtención de los datos, ya que los honorarios de los especialistas en esta materia son elevados (llegan hasta los 200 €/hora). Así, las empresas que han seleccionado una unidad específica para realizar las mediciones no ven fácilmente el retorno de la inversión que obtendrían si volvieran a realizar las mediciones sobre el mismo software con una unidad distinta. La consecuencia de todo esto es que la obtención de los datos procedentes de un entorno industrial es prácticamente imposible.

Para resolver este problema existen dos soluciones: o bien los investigadores, procedentes de otra institución —normalmente una universidad o centro de investigación— se encargan de medir con distintas unidades un conjunto de aplicaciones cedidas por la empresa en virtud de acuerdos subscritos por ambas organizaciones; o bien se pide a un conjunto de estudiantes que han finalizado sus estudios de un curso sobre estas unidades que, como proyecto final, realicen las mediciones necesarias con las distintas unidades estudiadas. En ambos casos el problema del coste de las mediciones se reduce muy significativamente.

Un segundo problema radica en la calidad de los datos obtenidos en las mediciones, cuyas consecuencias son dos: la realización de mediciones incorrectas, a causa de una formación inadecuada de los medidores, y la construcción de una base de datos heterogénea, debida a la inclusión en ella de mediciones realizadas por distintos medidores. Sobre este último aspecto no se ha publicado aún ningún trabajo que cuantifique el impacto que un determinado medidor tiene en la medida, si bien ya se están haciendo trabajos de investigación sobre este aspecto —entre ellos, uno por parte del equipo de Cuadrado-Gallego— que permiten llegar a unas conclusiones preliminares, al menos cualitativas, que establecen que existe un impacto en la medida imputable al medidor.

La solución a este segundo problema descansa en la participación de medidores expertos que, en el primer caso, realicen las mediciones y, en el segundo caso, revisen y hagan homogéneas las mediciones previas procedentes de varios medidores.

2.2. Metodología de obtención de los datos en la investigación actual

El procedimiento utilizado en este trabajo para la obtención de los datos ha sido diferente al usado en los trabajos previos. La forma de abordar la solución de los dos problemas descritos en el punto primero fue la siguiente: para la cuestión del coste de obtención de los datos se adoptó la segunda solución posible, de modo que estos se obtuvieron a través de los trabajos finales de estudiantes sobre medición, utilizando IFPUG y puntos de casos de uso de aplicaciones reales. Esta opción introducía el segundo problema, el de la calidad, tanto a través de la corrección de los datos —ya que los medidores no eran expertos—, como a través de la heterogeneidad, pues cada dato iba a ser obtenido por un medidor distinto.

El planteamiento para hacer frente a estos dos problemas tuvo una doble vertiente: en primer lugar, una rigurosa selección de los medidores entre los estudiantes que demostrasen un mayor conocimiento de la materia; en segundo lugar, la solución propuesta en el apartado primero, consistente en la revisión de todos los trabajos por parte de un medidor experto. A continuación se describen en detalle los pasos seguidos:

Enseñanza del proceso de puntos de casos de uso

Se dirige a estudiantes de la titulación de Ingeniería Técnica en Informática de Gestión, dentro de la asignatura obligatoria de tres créditos, de tercer curso, Laboratorio de Planificación y Gestión de Sistemas. Se imparte un total de 24 horas de clases teóricas

sobre los métodos IFPUG 4.1 y puntos de casos de uso, ocho horas de cada uno de los métodos, repartidas en clases de dos horas de duración. En dos grupos se impartió primero IFPUG y a continuación puntos de casos de uso; y en otros dos grupos se comenzó por puntos de casos de uso y posteriormente se trató IFPUG, con el objetivo de evitar los efectos que pudiera tener aprender antes unas unidades u otras. Las versiones elegidas de las unidades fueron IFPUG 4.1 y, en cuanto a puntos de casos de uso, la traducción al castellano del método de puntos de casos de uso de Gustav Karner.

Selección de los alumnos participantes en las mediciones

Para poder participar en la realización de las medidas, los estudiantes debían cumplir dos requisitos:

- Asistir al 90% de las clases; esto es, se podía faltar a una sola clase.
- Tener una nota superior a 8 sobre 10 en una prueba escrita de conocimientos, realizada en la undécima semana de clase (hicieron esta prueba todos los alumnos, tanto si iban a participar en las mediciones como si no lo iban a hacer).

Medición de una aplicación real con las unidades de puntos de casos de uso

A cada alumno seleccionado se le asignan las especificaciones de una aplicación real, distinta para cada uno, medida en cursos anteriores de la asignatura mediante el método IFPUG y que hubiera sido desarrollada previamente por la Universidad de Alcalá en la realización de algún proyecto de colaboración con una empresa y respecto a la cual la Universidad tuviera o compartiera la propiedad intelectual.

Cada alumno debe proceder a la medición de la aplicación asignada con las unidades de medida.

Corrección y uniformización de las mediciones hechas por los estudiantes

Un medidor experto revisa las mediciones de puntos de casos de uso realizadas por los estudiantes. Las tareas de revisión consistieron en: comprobar la existencia de errores en la identificación de los actores existentes en el sistema y en la identificación de los casos de uso; comprobar la existencia de errores en la identificación de las transacciones de los casos de uso; comprobar la existencia de errores en el cómputo de los puntos de actores, transacciones y puntos de casos de uso desajustados; comprobar la existencia de errores en la obtención de los factores de peso, de complejidad técnica y ambiental, y en el cómputo de puntos de casos de uso.

Los pasos descritos se aplicaron durante un curso, obteniéndose los siguientes resultados:

1. Generación de la muestra de puntos de casos de uso *mla08.*

- Se cuenta con un total de 74 alumnos, repartidos en cuatro grupos, tres de ellos tienen 18 alumnos por clase y uno tiene 20 alumnos.
- El número final de alumnos que cumplieron la norma de 90% de asistencia a clase fue de 63.
- De los 63 alumnos aptos para realizar las mediciones, superaron la prueba 56, con la siguiente distribución de notas: 16-aprobado, 23-notable, 11-notable con nota igual o superior a 8 y 6-sobresaliente.

Por lo tanto, se seleccionó a 17 alumnos para realizar las medidas.

2. Generación de la muestra IFPUG *jjcg06*.

- Se parte de un total de 83 alumnos, repartidos en cuatro grupos, tres de ellos de 20 alumnos por clase y uno de 23 alumnos.
- El número final de alumnos que cumplieron la norma de 90% de asistencia a clase fue de 71.
- De los 71 alumnos aptos para realizar las mediciones, superaron la prueba 63, con la siguiente distribución de notas: 18-aprobado, 34-notable, 10-notable con nota igual o superior a 8 y 11-sobresaliente.

Por lo tanto, se seleccionó para realizar las medidas a 21 alumnos.

3. Generación de la muestra de pruebas *jjcg06-mla08*.

Mediante la combinación de las muestras *jjcg06* y *mla08* se obtuvo una tercera muestra *jjgc06-mla08,* combinatoria de los métodos IFPUG y puntos de casos de uso, dado que los proyectos utilizados en la medición IFPUG efectuada en *jjcg06* fueron los usados para medir los puntos de casos de uso de la muestra *mla08*.

El campo de aplicación de *jjgc06-mla08* (C) se obtuvo mediante discriminación de un número de proyectos igual a la diferencia de muestras en *jjgc06* (P) y el número de alumnos seleccionados para realizar las mediciones en *mla08* (A).

$C = \min(A,P)$

$C = \min(17,21)$

$C = 17$

Fueron 17 los proyectos evaluados de manera satisfactoria, tanto con el método IFPUG como con el método de puntos de casos de uso.

3. Análisis de los datos

3.1. Búsqueda de las ecuaciones lineales

El primer paso del análisis que se realizó sobre los datos recogidos consistió en la búsqueda de las ecuaciones lineales que permitiesen la conversión entre los puntos de función IFPUG y puntos de casos de uso para cada una de las muestras.

1. Muestra *am04*

La ecuación lineal obtenida realizando un ajuste de mínimos cuadrados sobre sus 15 datos es:

UCP = 20,23 + 0,394 I

Con un R2 = 0,6.

2. Muestra *jjgc06-mla08*

La ecuación lineal resultante haciendo un ajuste de mínimos cuadrados sobre sus 17 datos es:

UCP = -0,62 + 0,94 I

Con un R2 = 0,78.

3. Muestra *am04- jjgc06-mla08*

La ecuación lineal obtenida haciendo un ajuste de mínimos cuadrados sobre sus 32 datos es:

UCP = -15,26 + 0,93 I

Con un R2 = 0,86.

La Figura 1 muestra gráficamente los conjuntos de datos utilizados y las ecuaciones de regresión obtenidas para ellos.

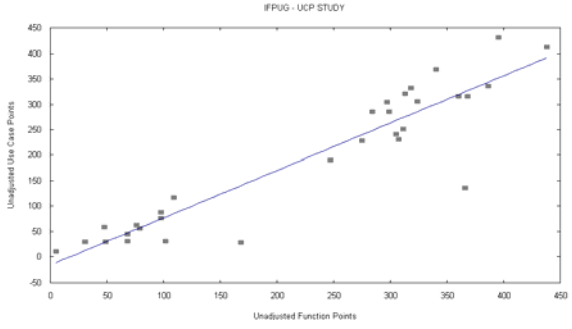


Figura 1. Ecuaciones lineales para *am04 -jjgc06-mla08.*

3.2. Análisis de las ecuaciones lineales

Un primer análisis de los indicadores estadísticos de las ecuaciones muestra que todos ellos están dentro de los márgenes considerados aceptables; si bien es cierto que los de la muestra *am04* están ligeramente fuera de los límites, las distancias no son suficientemente grandes como para despreciar los resultados obtenidos.

Cuando se analizan los parámetros de las ecuaciones se observa que las tres, aunque con diferencias notables en los términos independientes, muestran una tendencia marcada en las pendientes de las rectas, hacia el entorno de 1; si se toma la intersección, se obtiene el intervalo (0,7; 0,8). Esto muy significativo, ya que se puede observar la favorable evolución de la recta a mayor número de proyectos medidos con ambos métodos.

3.3. Búsqueda de las ecuaciones no lineales

Aunque en la bibliografía existente no se encuentra ningún estudio de la conversión de unidades de medición del tamaño funcional del software mediante el uso de ecuaciones no lineales, dos importantes razones indican que tal estudio podría proporcionar resultados interesantes:

- El análisis realizado utilizando ecuaciones lineales ha permitido constatar que tanto *jjcg06-mla08* como *am04-jjcg06-mla08* presentan una pendiente similar, en el entorno de uno, lo que parece indicar que sería posible encontrar un factor de conversión entre las unidades de medida IFPUG y puntos de casos de uso en ausencia de un término independiente.
- Un aspecto fundamental en la búsqueda del factor de conversión es el hecho de que la ecuación debería verificar el origen de coordenadas, ya que si hubiera un software con una medida de 0 IFPUG (es decir, sin tamaño), debería tener una medida de 0 puntos de casos de uso y en las ecuaciones lineales presentadas el tamaño que tendría sería el correspondiente al termino independiente. Este aspecto incluso se complica más cuando tal término independiente es negativo, ya que daría lugar a un software con un tamaño x, positivo, en IFPUG, que daría lugar no solo a

un tamaño 0 en puntos de casos de uso, sino a un tamaño negativo, lo cual es un absurdo conceptual.

Estas dos importantes razones llevaron, en el marco de las investigaciones presentadas aquí, a analizar el uso de ecuaciones no lineales en busca del factor de conversión. El nuevo análisis con ecuaciones no lineales se realizó, no solo para la muestra de datos obtenida en esta investigación, sino que también se aplicó a la muestra procedente de trabajos previos existentes. La Tabla 1 recoge los resultados obtenidos, una vez realizada la conversión inversa de los parámetros.

Estudio	**a**	**b**	**R^2**
am04	3,86	0,59	0,63
jjcg06-mla08	1,5	0,92	0,84
am04-jjcgg06-mla08	1,05	0,94	0,82

Tabla 1. Parámetros de las ecuaciones no lineales.

3.4. Análisis de las ecuaciones no lineales

Un primer análisis de los indicadores estadísticos de las tres ecuaciones muestra que todas ellas se sitúan dentro de los márgenes considerados estadísticamente aceptables.

Cuando se analizan los parámetros de las ecuaciones se observa que las muestras aceptadas presentan un comportamiento tendente, en el valor del exponente, hacia un entorno cercano a 1, lo cual es positivo para el estudio, dado que un exponente con estas características favorece una conversión unívoca entre dos unidades de medida. Igualmente, es destacable la tendencia en el valor del término independiente, el cual marca la proporcionalidad de la conversión. Esta tendencia es hacia la unidad, lo cual es muy significativo e interesante. Si se toma la intersección, analizando los intervalos más probables para el exponente, se obtiene el intervalo (0,7; 1,2), que es un resultado muy esperanzador, observándose que se ha acotado su dispersión respecto a la que se observaba para el término independiente de las ecuaciones lineales.

Si lo anteriormente expuesto se combina con los resultados específicos de la muestra *am04-jjcg06-mla08,* que es la que mayor número de datos tiene y, que en consecuencia, se podría considerar como la más significativa, y observando la manifiesta tendencia en el término independiente y en el exponente hacia su compensación en un entorno de 1, se puede apreciar que están en consonancia. Esto significaría que un punto de

función IFPUG equivaldría, aproximadamente, a un punto de función UCP, con un margen actual de precisión en un entorno de un 25%.

4. Conclusiones y trabajos futuros

Como resultado de las investigaciones realizadas, se ha podido concluir que el factor de conversión entre las unidades IFPUG y puntos de casos de uso está en un entorno en el que a un punto de función IFPUG corresponde un punto de casos de uso. No se ha concluido que sea una conversión exacta, sino aproximada. Así, en determinados casos, el rango de variación podría situarse en torno a un 25%. Se proponen los siguientes trabajos futuros: realizar nuevos análisis sobre nuevos conjuntos de datos más amplios, obtenidos con el doble objetivo de verificar los resultados de este estudio y reducir el intervalo de confianza para el factor de conversión, y contrastar los resultados obtenidos en este tipo de investigaciones, basadas en datos empíricos sobre la conversión de unidades de medición funcional del software, con las que se están llevando a cabo y que se basan en análisis cualitativos.

Referencias

[1] Albrecht, A. J., "Measuring Application Development Productivity", en *Proceedings Joint SHARE, GUIDE and IBM Application Development Symposium*, IBM, pp. 83-92, 1979.

[2] Capers, J., "Backfiring: Converting Lines-of-Code to Function Points", *Computer*, vol. 28, nº 11, pp. 87-88, 1995.

[3] Jacobson, I., *Object-Oriented Software Engineering*, Addison Wesley Professional, 1992.

[4] Karner, G., Use Case Points. Resource Estimation for Objectory Projects, Objective Systems SF AB, 1993.

Revista Española de Innovación, Calidad e Ingeniería del Software, Vol.4, No. 2, 2008

Making Software Process Management Agile

José Manuel García, José Javier Berrocal, Juan Manuel Murillo
Universidad de Extremadura
{jgaralo,jberolm,juanmamu}@unex.es

Abstract

The software development process is becoming more complex everyday and thus requires specific tools that facilitate its management. In this article, Zentipede, a tool for managing the software development process in software factories, is presented. This tool simplifies the software process management and provides a lot of statistical data to help measure and improve the quality of developments carried out. Real data obtained in a software factory and the results after using the tool are also presented.

Key words: software process, development process, software factory.

Resumen

El proceso de desarrollo software es cada día más complejo y por lo tanto requiere de herramientas cada vez más especificas que faciliten su gestión. En este articulo se presenta Zentipede, una herramienta para la gestión del proceso de desarrollo software en entornos de fábricas de software. La herramienta simplifica la gestión del proceso software y proporciona una gran cantidad de datos estadísticos que ayudan a medir y mejorar la calidad de los desarrollos llevados a cabo. Se presentan también datos reales obtenidos en una fábrica de software y los resultados conseguidos tras la utilización de la herramienta.

Palabras clave: proceso software, proceso de desarrollo, fábrica de software

1. Introduction

In the last few years, software development has been directed towards obtaining higher quality products in the least possible time and at the lowest cost. One of the main steps applied to achieve these objectives has consisted in discarding obsolete waterfall development processes and replacing them with more agile and iterative processes [1].

The adoption of this type of development process along with the constantly expanding tendency to centralize development in software factories, with its benefits and disadvantages [2], has increased the need for tools that improve the control and exploitation of the resources. On one hand, tools are needed to measure the use of the development processes and the quality they provide (http://kpilibrary.com/), [3]. Tools are also needed to facilitate the management of the software process in this kind of environment [4].

Although there are tools that cover some of these characteristics, many of them are not adapted to agile and iterative development processes. Moreover, the integration of many tools is clearly needed in order to avoid the duplication of effort [5], [6].

In this paper, we present Zentipede[4] [5] (http://www.zentipede.org/), a tool designed to fulfill these needs. This system facilitates the management of the software development process in distributed environments supporting collaborative work in software development. This tool automates some tasks for quality management, provides statistical data about the quality of developments and integrates the functionality of multiple applications to reduce the duplication of effort.

The system originated from a real need present in a software factory located in the University of Extremadura. In this environment, we detected some deficiencies in software process management in existing applications, mainly because they did not focus on agile and iterative processes.

The second section of this paper describes what motivated the creation of the tool. In the third section its main features are presented, giving more detail about every one of the modules that compose it in sections four, five and six. Its advantages over other related work are detailed in section seven. The results obtained after its use in a software factory are discussed in section eight. The last section deals with conclusions and future work.

2. Motivation

The tool Zentipede arose, as mentioned in the previous section, to fulfill a need present mainly in software factories. The specific case at hand is that of the Teseo and InodUex factories created at the University of Extremadura with the collaboration of SDAE and INDRA, respectively. In these factories, the projects developed were mainly sent by the founding companies from some of their different locations.

Due to this relocation of development and developers, there was an increase in the need for exhaustive controls that allowed for the best possible use of available resources at all times. Typically, these management tasks are done by hand.

[4] The name of the system comes from the English word centipede. Zentipede has been chosen as a hallmark and due to the similarity of the Z with a moving centipede.

[5] The project has been financed with funds from CICYT project TIN2005-09405-C02-02 and tested in the software factories Teseo and InodUEx.

As the factory grew and the number of projects and resources involved increased, the time necessary for the management tasks also increased. This produced an increased workload for managers, which led to less time allocated to the development of projects in order to increase the time spent managing them.

Zentipede was conceived as a tool to fulfill these needs and facilitate the management of software development processes carried out by relocated teams. To do so a web application was developed first to collect information from everyday tasks carried out in a software factory and generate reports from them.

3. General characteristics

The main aim of the Zentipede tool is to facilitate the management of the development process in a software factory. To fulfill this function the tool must adapt to the software process used by the factory but be flexible enough to allow for variations required by customers. From this starting point, Zentipede provides a wide range of features. These characteristics are grouped according to the functions they fulfill.

On one hand, we have the following features whose aim is to optimize the use of the resources. The tool allows for a complete management of the workers and their workday. The tool can also collect information about all the projects. From this information it is possible to manage the participation of workers in the different projects and maintain a complete control of the number of hours spent by each of them on each project. Finally, it can control the unavailability of workers.

Moreover, the system has another series of characteristics to improve the quality of the software development process. Each project is divided into use cases and these, in turn, are divided in more fine-grained tasks, simpler to carry out and estimate their length and complexity. It is possible to assign these fine-grained tasks to one or more workers during the different iterations of the project so as to keep control of the part of the project in which a particular worker actively participates in each iteration. That control may be done in a more exhaustive way; it is even possible to associate both tasks and workers with fragments of source code. Finally, the system generates a large number of reports in different formats from the information stored in it. These reports are very useful to evaluate various aspects of a software factory, from a global level to that of a specific project or worker.

To conclude, the system has one last feature aimed at minimizing duplication of efforts and automating part of the process. To avoid the duplication of work the system helps generate the documentation of a project from the information it has collected. This is achieved through the use of a wiki as a collaborative repository for documentation of projects.

In addition to these functions, one of the key concepts that has dominated the creation of Zentipede is the simplicity of use. Throughout the entire development, the intention has been to minimize, as much as possible, the time required for data entry because this time is subtracted from that available for project development.

The following sections describe the various modules that make up Zentipede and which provide all the functions.

4. Zentipede Web Portal

In this section the most important module of Zentipede is described. This module is responsible for providing most of the functionality described in the previous section. The rest of modules that make up the tool are complements of this.

This web application is responsible for collecting data from the daily work done by all the members of the software factory. From these data a series of reports are generated, which give us a lot of information about the quality process being followed, especially of the areas of project planning and project monitoring and control described by CMMI[6].

The operation of the tool for the three roles involved in it is described below.

4.1 Developer

The application presents to the developer the task that he must perform daily. At the end of the day the developer should indicate the progress made and the hours spent on each task. This way a large amount of information about the work being done in a factory is obtained almost without interfering with the work of development.

4.2 Project manager

The users of this profile are the ones who must provide the most data to the tool. The application provides these users with the same functionality as that provided to developers

[6] CMMI. http://www.sei.cmu.edu/cmmi/models/index.html

and thus a part of the information to be provided is the same, specifically the time spent in the realization of the various tasks that they carry out.

In addition to this basic functionality, project managers have greater control over one or several of the projects being developed in the factory. This control is carried out through a workflow consisting of the steps shown below. For a better understanding of the workflow, Figure 1 shows it in the form of a business process. For greater clarity, tasks carried out by developers have been included in the business process.

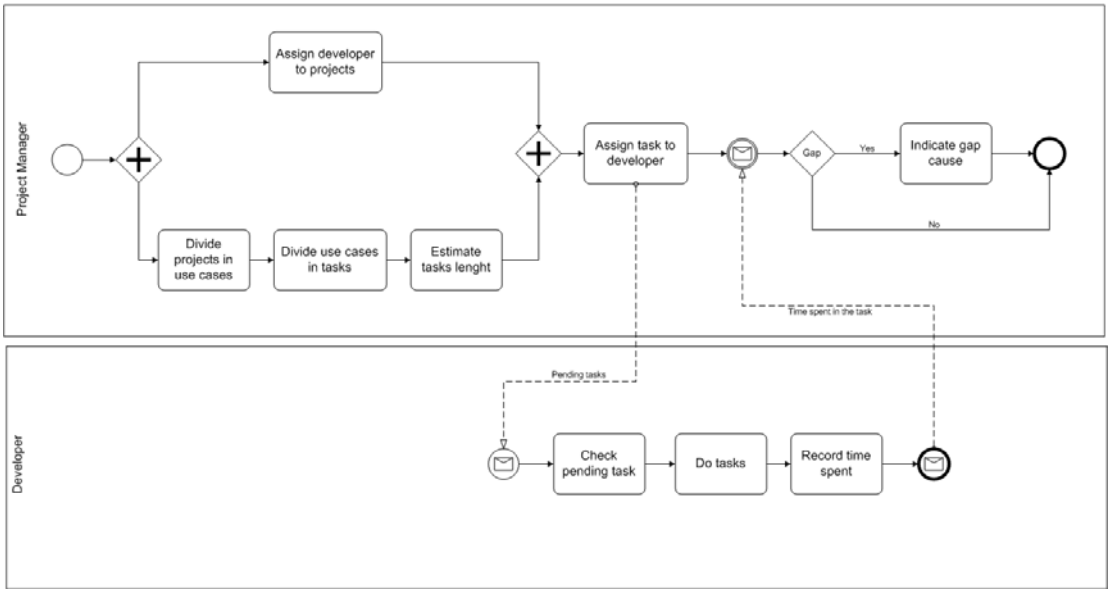


Figure 1. Business process followed by project managers.

The time that project managers must invest in the use of the tool is, as can be seen, considerably more than the time spent by the developer. But this dedication is still less than the time spent on management tasks before this tool. Moreover, the results are better than those that similar tools provide, as may be observed throughout this article.

4.2 Administrator

These users are those who benefit the most with the use of this application. They have at their disposal the same functionality as the project managers, expanded with a wide range of possibilities that give them complete control over the tool and the projects that are managed with it. These users also get lots of reports about the status of the factory and all the projects and workers involved at all times.

For the management of a project using this tool the only task that can only be done by the administrators, as such, is the inclusion of the project itself in the system. Once a project is created the rest of the necessary information can be provided, as has been seen, by project managers and developers.

Moreover, these users have a number of functions to modify some aspects of this tool to suit their specific needs.

But the biggest advantage of this application is the wide variety of data and reports it provides to administrators. These reports are obtained from all information provided. These reports also have the great advantage that many of them are connected to each other, allowing us to jump from one to another comfortably and thus find necessary information easily. Some of these reports are shown in later sections.

Administrators have all this information to improve the management of the development process and achieve a better use of available resources. Likewise, better control of the software process is achieved, which implies an improvement in quality.

5. Zentipede Eclipse Plug-in

This module was designed primarily to meet two goals. Its main function is to provide information concerning the code which the developers are working on. Therefore, it is possible to relate the tasks carried out by workers with the part of the code affected by these tasks. The other function provided by this module is to facilitate the daily use of the tool.

To achieve these objectives the most logical solution is to concentrate on the development environment. Because of this, this module was developed in the form of a plug-in for the Eclipse development environment[7]. This environment was chosen because it is one of the most used IDEs today, its architecture is perfectly prepared for the inclusion of plug-ins and it is licensed as FOSS.

The plug-in is connected with the web application, based on SOA, and the list of tasks assigned to the user is obtained. This task list is shown to the user. Once the assigned tasks are obtained, it is possible to register, in the system, the time spent in developing each of them from the development environment. This achieved one of the objectives of this

[7] Eclipse. http://www.eclipse.org

module, facilitate the daily use of the tool, so much so that developers can completely avoid the connection with the web application.

The mechanism for the registration of the changes to the code is more complex. The way to achieve this is to select, from the list of assigned tasks, the one the user is working on. By selecting one of the pending tasks the plug-in is responsible for monitoring the development environment to register the code that is changing.

This monitoring differs according to the files the user is working with. For files that do not correspond to Java code the only information that is recorded is that the user is modifying them. However, if it is a Java code file the environment provides a lot of information. When the user works with Java code files, the plug-in not only records the file that is being manipulated but also keeps a record of the packages that are being imported and classes that are changing within the file. Also, within a given class, the plug-in records what attributes or methods the user is working on.

Thanks to this monitoring, very precise information is obtained about the code areas in which the tasks are implemented and what users have done so. All this information is stored in the Zentipede Web Portal module and can be consulted later. This contributes very positively to the traceability of the product and the process and ultimately to the quality of both. Besides that, these data have another utility: when a user receives his list of pending tasks he also receives the areas of source code associated with them that have been modified during the realization of this task, so that a user who is assigned to a task can quickly find out what parts of the code other people have worked on.

6. Zentipede Documentation Center

In this section, the last module of Zentipede is presented. This module was designed to try to reduce one of the most common drawbacks in the development process: the duplication of data and the potential for inconsistencies.

A situation that occurs very frequently during the development process is the need to enter the same data in the different tools that are being used, with the risk of inconsistencies that this entails. The module that we present in this section is intended to mitigate, as far as possible, this problem.

In particular, this module tries to reduce these problems. To do so, the main web application has been integrated with a wiki whose main mission is to collect all the documentation of the project. The implementation of wiki chosen to integrate into Zentipede has been MediaWiki[8] because it is widely used and has a free software license. This integration has been done with the intention to reuse the information entered in the web application to be included automatically in the documentation associated with the project.

To integrate this module with the Zentipede Web Portal module, the Spring framework facility for aspect programming has been used. In this way, it is possible to incorporate the new functionality in a completely transparent way.The features added by this module are detailed below.

It is possible to define the desired structure for the documentation of the projects without being subject to any restrictions. A structure of documentation is associated to each project within the tool. This structure is generated automatically on the wiki.The structure of the documentation may be associated with project maintenance actions carried out in Zentipede Web Portal. So when one of these actions occurs in the web application, it affects to the documentation in the wiki. Because the wiki is accessible through the Web, the entire generated repository can be accessed directly, without using the main module. Taking advantage of the capabilities offered by MediaWiki, a historical trace of all actions carried out on the documentation of projects is maintained.

Through the use of this module, it is therefore possible to maintain the documentation of each project, with the necessary structure in each case, in a repository that allows for collaborative editing while taking advantage of the information entered in the main module to complete this documentation automatically.

7. Related work

There are other tools similar to Zentipede. In this section some of the best known ones, whose deficiencies have led to the creation of this system, are presented along with their advantages and disadvantages compared to our application.

[8] MediaWiki. http://www.mediawiki.org/

In general, most of these tools have two disadvantages, the first of which is that they are not oriented to agile and iterative development processes and, mainly, they have not been designed for use in software factories. The second disadvantage is that they focus their functionality on a single aspect of the development process, dissociating them from other aspects.

One of these applications that stands out is dotProject[9], one of the most widely used tools. However, it not only shows the disadvantages already mentioned, but also has problems with obtaining statistics about the information collected, which make it less useful for the management of an environment like a software factory.

Two other interesting tools in this field, although more limited than the previous one, are activeCollab[10] and BaseCamp[11]. However, they present a serious inconvenience: they are not licensed as free software and therefore cannot be easily expanded with the functionality needed.

Finally, there are other tools, such as LightHouse[12] and CruiseControl[13], but they are focused on more specific aspects of the development process, such as the management of the bugs that appear in the projects or the building process. This specification, however, introduces features that could be very interesting for the management of the development process.

8. Validation

The tool that is presented in this article has been used successfully in a software factory consisting of approximately 30 people. Throughout the period of use, more than ten projects were developed. The tool has recorded data from daily work in all these projects, so that after the completion of the tests the system has information on more than 7500 man-hours as shown in Figure 2.

[9] DotProject. http://www.dotproject.net/
[10] ActiveCollab. http://www.activecollab.com/
[11] Basecamp. http://www.basecamphq.com/
[12] LightHouse. http://lighthouseapp.com/
[13] CruiseControl. http://cruisecontrol.sourceforge.net/

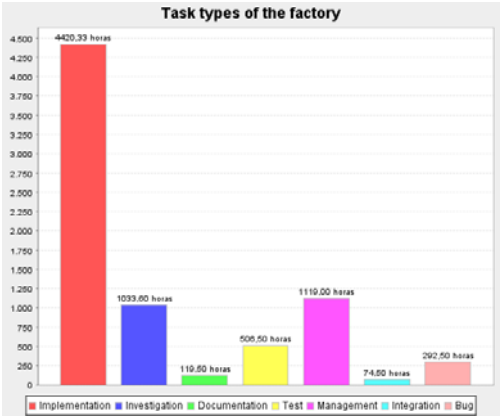


Figure 2. Worked hours during the use period of the tool, classified by task type.

The figure shows the most common type of task in a software factory is, as might be expected, the implementation tasks. However the large number of hours spent on management tasks must be highlighted, as this demonstrates the need for a tool like Zentipede, which facilitates these tasks and helps increase productivity, in such an environment. Two curious aspects stand out in this figure. The first one is the large number of hours spent on research task motivated by the location of the factory in a university environment. The other one is the little time spent on tasks of integration or correction of errors, which was due to the fact that much of these tasks were performed by the founder company of the factory in its other locations.

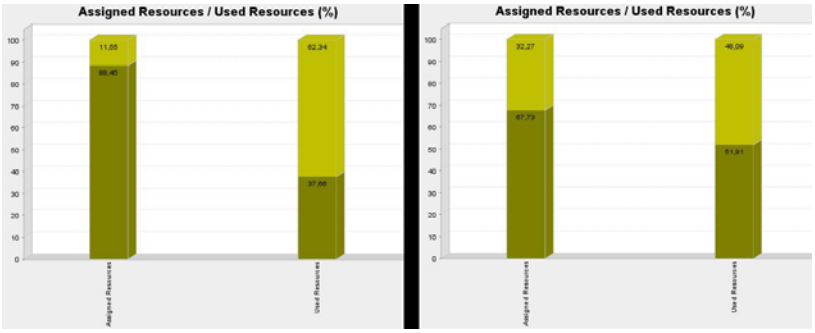


Figure 3. Data about the assigned resources compared to the used resources and their evolution.

The results have been clearly positive. The time spent on the management of the software process has been significantly reduced, while the information related to the process followed for each of the projects has increased remarkably. In addition, thanks to

the statistics provided by the tool, the accuracy of the time estimates increased and thus the utilization of available resources improved. This increase in the accuracy of the estimates is evident in Figure 3. This figure shows two equal graphs of the assigned resources and the resources actually used. The graph on the left is one of the first few weeks of use of the tool. In this graph the differential between resources allocated to those used is of more than 50 points. In the second graph, of the subsequent period after the experience gained through the use of the tool, the differential is reduced to less than 20 points.

These improvements in the estimates are largely due to information provided by the tool itself. In Figure 4, the difference between the estimated time and worked time on several projects is shown. This gap is further divided into positive gap (more time than needed was estimated) and negative gap (less time than needed was estimated).

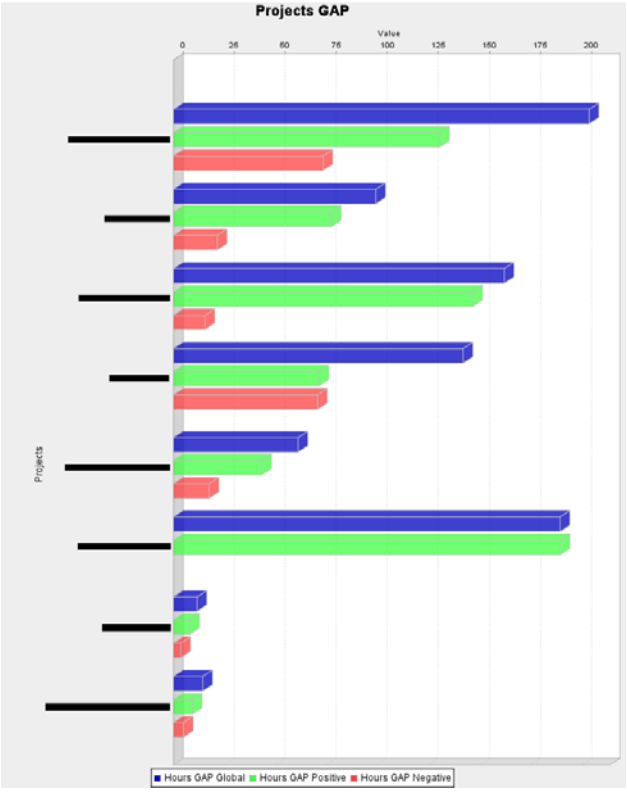


Figure 4. Difference between estimation work and realized work classified by project

The benefits are not limited to a better use of available resources or a decrease in the time spent on management of the software process. In addition, thanks to the use of Zentipede, the factory has increased its database, with detailed data on the development of several projects. Such information will serve to take on new projects with more exact

criteria about the capacity and productivity of the factory. An example of this can be seen in figure 5[14] where number of assigned resources to several projects over a period of time is shown. These data can help managers to decide about the number of resources a new project will require or the ability of the factory to deal with new developments.

9. Conclusion and future work

In this paper, we have presented the Zentipede system, the motivations that led to its creation, its main characteristics and the different modules that compose it. Also, a study of existing systems that cover part of the same functionality has been done, paying special attention to the benefits of Zentipede. In conclusion, details of the results obtained from the use of the system in real software factories have been given.

The current state of the system corresponds to the one described in this article, however it is constantly being improved. The inclusion of the support for new programming languages, among those supported by the environment like C + +, is being considered for Zentipede Eclipse Plugin. The ability to connect Zentipede Web Portal through SOA is also being expanded. In addition, the Zentipede system is part of a much larger project in progress. The final system aims not only to cover the management of the software process, but also to assist and automate, as much as possible, its execution at all levels, from capturing requirements to obtaining source code.

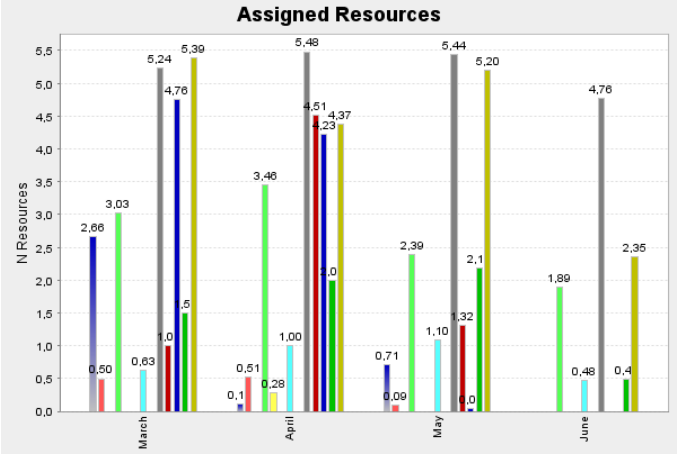


Figure 5. Assigned resources to different project during a period of time

[14] Projects names omitted for confidentiality purposes in figures 4 and 5.

Finally the authors want to highlight the availability of the system as free software and their disposal to address any doubts that may arise about the system or suggestions on possible improvements for future versions.

References

[1] Larman C. *Agile & Iterative development: a manager's guide*. Addison Wesley, 2005.

[2] Prikladnicki, R., Audy, J.L.N., Damian, D., y de Oliveira, T.C. "Distributed Software Development: Practices and challenges in different business strategies of offshoring and onshoring", en: Paulisch, F., Kruchten, P. y Mockus, A. (eds.) *Proceedings of the Second IEEE International Conference on Global Software Engineering 2007*, pp. 262-274, 2007.

[3] Dunn, R. N., "KPIs for Agile Teams", *Agilejournal*, Vol. 3 Num.5, 2008, www.agilejournal.com (consultado en julio de 2008)

[4] Ambler S.W. y Nizami, K. "Agile Strategies for Geographically Distributed Quality Management", *Agilejournal*, Vol. 2 Num.11, 2007. www.agilejournal.com (consultado en julio de 2008)

[5] Sinha, V., Sengupta, B. y Gosal, S., "An adaptive tool integration framework to enable coordination in distributed software development" en: Paulisch, F., Kruchten, P. y Mockus, A. (eds.) *Proceedings of the Second IEEE International Conference on Global Software Engineering 2007*, pp. 151-155, 2007.

[6] Krishnamurthy, V., "Benefits Of Tool Integration In Distributed Agile Development". *Agilejournal*, Vol. 1 Num. 6, 2006. www.agilejournal.com (consultado en julio de 2008)

La norma ISO/IEC 25000 y el proyecto KEMIS para su automatización con software libre

José Marcos[1], Alicia Arroyo[1], Javier Garzás[1 y 2], Mario Piattini[3]
[1]Kybele Consulting SL
{Jose.Marcos, Javier.Garzas, Alicia.Arroyo}@kybeleconsulting.com
[2]Grupo de Investigación Kybele. Universidad Rey Juan Carlos
Javier.Garzas@urjc.es
[3]Grupo Alarcos. Dpto. de Tecnologías y Sistemas de Información.
Universidad de Castilla-La Mancha
Mario.Piattini@uclm.es

Abstract

The quality control should be desirable done form a quantitative point of view, for which it is necessary to establish measurement systems throughout the product lifecycle. Code metrics provide the basis for the improvement and, in particular the product, provide direct visibility of product quality. However, in order to do efficient the quality systems, they should have a high level of automation and provide information with varying degrees of detail to the different actors surrounding the product. With these purposes, this paper presents technological and methodological measurement infrastructure that can be executed in a specific way or in an environment of continuous integration and which is based on the standard ISO/IEC 9126, providing different levels of product quality information.

Key words: quality, measurement, quality control panels.

Resumen

El control de la calidad se debe realizar desde un punto de vista cuantitativo, para lo cual es necesario establecer sistemas de medición durante todo el ciclo de vida del producto. Las métricas de código proporcionan la base para la mejora; en concreto, las de producto proporcionan visibilidad directa de la calidad del producto. Sin embargo, para que los sistemas de calidad sean eficientes, deben tener un alto grado de automatización y proporcionar información con diferente grado de detalle a los diferentes actores que rodean al producto. Con este objetivo, este artículo presenta infraestructura tecnológica y metodológica de medición que se puede ejecutar de forma puntual o en un entorno de integración continua, y que se basa en el estándar ISO/IEC 9126, proporcionando de esta forma diferentes niveles de información para la calidad del producto de software.

Palabras clave: calidad, métricas, cuadros de mando de calidad.

1. Introducción

"Quality is a complex and multifaceted concept." La calidad puede describirse desde diferentes perspectivas [1]; en el desarrollo del software hay dos especial y

tradicionalmente importantes: la calidad del producto en sí y la calidad del proceso para obtenerlo (o actividades, tareas, etc., para desarrollar y mantener software). Dos dimensiones esenciales, estudiadas desde hace tiempo [2] y que giran e interactúan en torno a la idea de que la calidad del producto está determinada por la calidad del proceso usado para desarrollarlo y mantenerlo.

No obstante, hoy en día parecen estar más de moda los modelos de mejora de procesos (CMMI, ISO 12207 e ISO 15504, principalmente), como reflejan recientes informes [3, 4] que, aunque incluyen actividades de medición de procesos y productos, se centran en los procesos. Sin embargo, como hace tiempo comentaban Kitchenham y Pfleeger[5], la principal crítica a esta visión es que hay poca evidencia en que cumplir un modelo de procesos asegure la calidad del producto; además, la estandarización de los procesos garantiza la uniformidad en la salida de estos, lo que puede incluso institucionalizar la creación de malos productos. Más recientemente, Maibaum y Wassyng [6], en esta línea, comentaban que las evaluaciones de calidad deberían estar basadas en evidencias extraídas directamente de los atributos del producto, no en evidencias circunstanciales deducidas desde el proceso.

En este artículo se muestra una implementación de la parte referente a la mantenibilidad de la norma ISO/IEC 25000 [7], haciendo uso de herramientas de software libre, lo que permite obtener una medida de la calidad del producto de software. El entorno, denominado KEMIS (Kybele Enviroment Mesaurement Information System), proporciona una infraestructura para dicha medición y que se puede ejecutar de forma puntual o integrada en entornos de integración continua, permitiendo obtener de forma automática y periódica un conjunto de informes relativos a la calidad del producto, obteniendo métricas de código y microarquitectura, dentro del marco de la norma 25000. Las restricciones de KEMIS vienen dadas por la parte de la mantenibilidad que se incluye en la división 2502n. Además, varias empresas importantes (empresas públicas, una administración regional y una administración pública, entre otras) están usando este entorno para la evaluación de productos de software.

En el epígrafe segundo se tratan las normas ISO encargadas de la calidad del producto de software; en el apartado tercero, las métricas de producto y su relación con las herramientas de software libre; en el cuarto, el proyecto KEMIS y su relación con la calidad

del producto; en el apartado quinto se presenta un ejemplo de evaluación y, por último, en el apartado sexto se ofrecen las conclusiones.

2. Las normas ISO para la calidad de un producto de software

En el año 1991 la ISO (International Organization for Standardization) publicó su modelo de calidad para la evaluación del producto de software (ISO 9126:1991), que fue extendiendo con revisiones hasta 2004, dando lugar a la actual norma ISO/IEC 9126 "Software Engineering. Product Quality" [8]. La norma ISO/IEC 9126 propone un conjunto de características, subcaracterísticas y atributos para descomponer la calidad de un producto de software. Propone seis propiedades (funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad), que se dividen en subcategorías, como se muestra en la Figura 1.

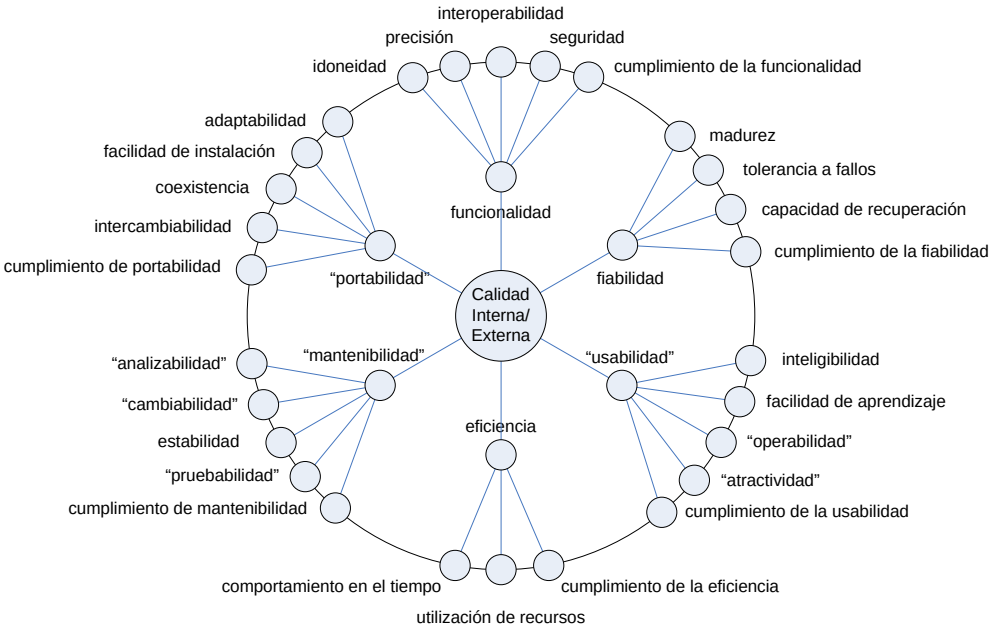


Figura 12. Características de la calidad interna y externa según la ISO/IEC 9126.

Recientemente ha aparecido una nueva versión de la norma 9126: la norma ISO/IEC 25000. Esta proporciona una guía para el uso de las nuevas series de estándares internacionales, llamados Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE). Constituyen una serie de normas basadas en la ISO 9126 y en la ISO 14598 [9] (Evaluación del Software), y su objetivo principal es guiar el desarrollo de los productos de

software con la especificación y evaluación de requisitos de calidad. Establece criterios para la especificación de requisitos de calidad de productos de software, sus métricas y su evaluación. Incluye un modelo de calidad dividido en dos partes para unificar las definiciones de calidad de los clientes con los atributos en el proceso de desarrollo. SQuaRE está formada por las divisiones siguientes:

- ISO/IEC 2500n. División de gestión de calidad. Los estándares que forman esta división definen todos los modelos comunes, términos y referencias a los que se alude en las demás divisiones de SQuaRE.
- ISO/IEC 2501n. División del modelo de calidad. El estándar que conforma esta división presenta un modelo de calidad detallado, incluyendo características para la calidad interna, externa y en uso.
- ISO/IEC 2502n. División de mediciones de calidad. Los estándares pertenecientes a esta división incluyen un modelo de referencia de calidad del producto de software, definiciones matemáticas de las métricas de calidad y una guía práctica para su aplicación. Presenta aplicaciones de métricas para la calidad de software interna, externa y en uso.
- ISO/IEC 2503n. División de requisitos de calidad. Los estándares que forman parte de esta división ayudan a especificar los requisitos de calidad. Estos requisitos pueden ser usados en el proceso de especificación de requisitos de calidad para un producto de software que va a ser desarrollado ó como entrada para un proceso de evaluación. El proceso de definición de requisitos se guía por el establecido en la norma ISO/IEC 15288.
- ISO/IEC 2504n. División de evaluación de la calidad. Estos estándares proporcionan requisitos, recomendaciones y guías para la evaluación de un producto de software, tanto si la llevan a cabo evaluadores, como clientes o desarrolladores.
- ISO/IEC 25050–25099. Estándares de extensión SQuaRE. Incluyen requisitos para la calidad de productos de software "*Off-The-Self*" y para el formato común de la industria (CIF) para informes de usabilidad.

La norma ISO 25000 ha sido desarrollada por el subcomité SC 7 (Ingeniería de software y sistemas) del Comité Técnico Conjunto ISO/IEC JTC 1.

3. Las métricas de producto y su medición con software libre

Los citados modelos de calidad del producto se basan en el concepto de métrica como base para su aplicación. En lo que se refiere a métricas, una de las más representativas es son las Líneas de Código (LOC, Lines of Code).[10] Otra métrica de especial relevancia es la Complejidad Ciclomática (CC, Cyclomatic Complexity), definida por McCabe[11], que se utiliza para evaluar la complejidad de un programa. Desde 1976 ha aparecido una gran cantidad de métricas, entre las que podemos destacar las definidas por Chidamber y Kemerer[12] o las que establecen Brito e Abreu y Carapuça[13], que se usan dentro del paradigma de Orientación a Objetos.

Un importante avance en este sentido es que en la actualidad existe un amplio abanico de herramientas de software libre que permiten obtener dichas métricas de producto. Entre ellas destacan JavaNCSS (www.kclee.de/clemens/java/javancss), PMD (http://pmd.sourceforge.net), Checkstyle (http://checkstyle.sourceforge.net), Findbugs (http://findbugs.sourceforge.net) y JDepend (www.clarkware.com) que, junto con herramientas de construcción de software como Maven (http://maven.apache.org/) y Ant (http://ant.apache.org/), y de integración continua como Cruise Control (http://cruisecontrol.sourceforge.net/) o Continuum (http://continuum.apache.org/), permiten automatizar la obtención de las métricas, personalizar los resultados y definir la periodicidad con la que se realizan las mediciones.

4. El proyecto KEMIS: un entorno de software libre para la medición de la mantenibilidad del producto

"En ocasiones la proporción de coste que supone el mantenimiento ronda el 90%"[14]. El mantenimiento en el ciclo de vida del software se considera el proceso con más peso económico, debido a que se basa en la labor de corrección de errores durante un largo período de tiempo y en la facilidad con la que se puede añadir funcionalidad. Cabe destacar que el mantenimiento del producto está estrechamente relacionado con la fiabilidad del sistema (el tiempo de corrección de un fallo determina el tiempo en que no se puede usar una funcionalidad o un sistema completo), que es algo que incide directamente sobre el usuario y el cliente.

La aportación principal de KEMIS es ofrecer indicadores de mantenibilidad que aporten visibilidad durante el desarrollo, la adquisición y el mantenimiento de un producto, proporcionando una infraestructura de medición automática, que se basa en métricas que se pueden obtener con herramientas de software libre y el modelo de calidad que propone la ISO/IEC 9126.

4.1. Atributos de calidad

El modelo de calidad de la ISO/IEC 9126 establece que las características están compuestas de subcaracterísticas y estas, a su vez, de atributos de calidad.

Los atributos cobran especial relevancia, ya que son el punto de unión entre el modelo de calidad y las métricas disponibles, en especial, las más maduras y ampliamente aceptadas, que hoy en día se pueden obtener con herramientas de software libre.

El modelo se basa, por tanto, en la definición de atributos de calidad que se puedan obtener directamente del producto. En KEMIS no solo se ha realizado una selección de atributos de calidad relativos a subcaracterísticas de la mantenibilidad, sino que se han establecido las funciones necesarias para obtenerlos a partir de métricas de código.

Característica	**Subcaracterística**	**Atributo**
Mantenibilidad	Capacidad para ser analizado	Densidad de complejidad ciclomática Densidad de código repetido Densidad de comentarios Densidad de defectos de capacidad de ser analizado
	Capacidad para ser cambiado	Densidad de defectos de facilidad de cambio Densidad de dependencias cíclicas
	Estabilidad	Densidad de defectos de estabilidad
	Capacidad para ser probado	Densidad de defectos en pruebas unitarias Densidad de pruebas unitarias Cobertura de pruebas unitarias Tasa de fallos de pruebas unitarias Tasa de errores fe pruebas unitarias
	Cumplimiento de la mantenibilidad	Densidad de defectos de mantenibilidad definidos por la empresa

Tabla 3. Relación de atributos contemplados en KEMIS.

En la Tabla 1 se recoge la selección de atributos de calidad realizada y las subcaracterísticas de calidad a las que afectan.

4.2. Normalización de atributos de calidad

Los atributos de calidad reflejan propiedades heterogéneas del producto de software; por eso, al manejarlos para componer una subcaracterística, es deseable que estén normalizados y que sus posibles valores pertenezcan a los mismos rangos con el mismo significado.

Una de las aportaciones de KEMIS es que proporciona funciones para normalizar los indicadores de atributos de calidad, de tal forma que se permite su escalado a subcaracterísticas y características de calidad, además de facilitarse su interpretación. En este sentido, antes de pasar a componer una subcaracterística, los atributos se pasan por una función de calidad cuyo objetivo es proporcionar un valor de calidad entre 0 y 100.

Densidad de complejidad ciclomática	
Nivel	Táctico
Modelo de análisis	$DenCC = \frac{\sum CC}{NCSS} \times 100$
Atributos	**DenCC:** Densidad de la complejidad ciclomática **CC:** Complejidad ciclomática **NCSS:** Non Commenting Source Statements (sentencias de código fuente que no son comentario)
Función de nivel de calidad	$Calidad(DenCC) = \begin{cases} 0 & ,si \quad DenCC \le m \\ \frac{DenCC - m}{n - m} \times 100 & ,si \quad m \le DenCC \le n \\ 100 & ,si \quad n \le DenCC \le u \\ \frac{v - DenCC}{v - u} \times 100 & ,si \quad u \le DenCC \le v \\ 0 & ,si \quad DenCC > v \end{cases}$
Umbrales recomendados	m = 0,08; n = 0,16; u = 0,24 y v = 0,36
Criterio de calidad	Los valores inferiores al umbral *m* o superiores al umbral *v* serán considerados como nulos. Los valores que sean mayores que el umbral *n* y menores que el umbral *u* serán considerados como óptimos. El resto de valores serán proporcionales a la distancia entre los valores nulo y óptimo.
Interpretación del indicador	Los valores cercanos a 0 indican un nivel deficiente de calidad en el atributo. Los valores cercanos a 100 indican un valor alto en él.

Tabla 4. Normalización de la densidad de complejidad ciclomática.

En la Tabla 4, a modo de ejemplo, se muestra una definición del atributo Densidad de complejidad ciclomática: la forma de calcularla, su función de calidad, los criterios que se han adoptado para conseguir esa función y los umbrales asociados.

La función de calidad resultante se puede ver en la Figura 2.

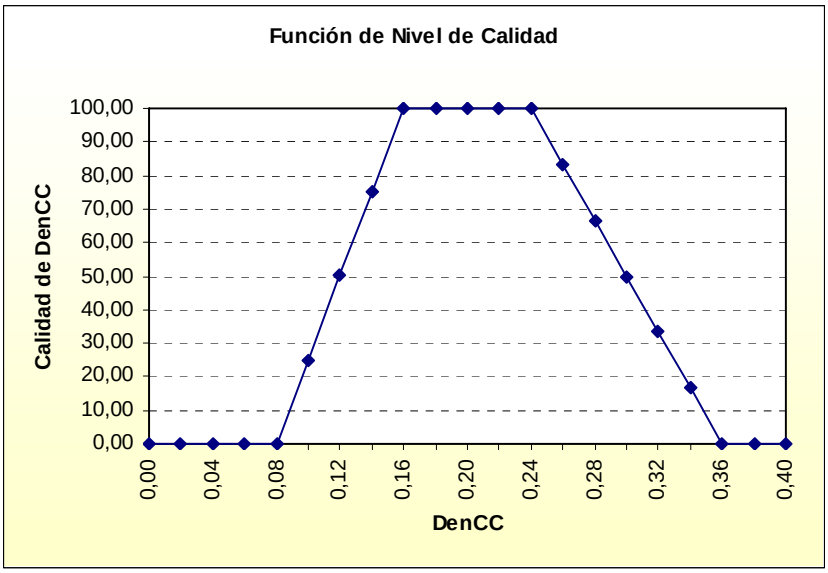


Figura 2. Función de calidad de la densidad de la complejidad ciclomática.

Esta función permite normalizar atributos de calidad. Mientras que para la Densidad de la complejidad ciclomática podemos obtener valores que oscilan en un rango entre 0 y 1, existen otros atributos que pueden tener otros rangos; por ejemplo, entre 0% y 100%, por lo que es necesaria una normalización de cada atributo.

4.3. Escalado de indicadores de calidad

En KEMIS, una vez definidos y normalizados los atributos, se hace uso de unas funciones que permiten componer las subcaracterísticas, dando mayor importancia a unos atributos de calidad que a otros, asignando pesos.

En la Tabla 5 se muestra cómo se hace la composición de una subcaracterística, la Capacidad para ser analizado, a partir del conjunto de atributos que se muestran con anterioridad en la Tabla 3.

Capacidad de ser analizado	
Nivel	Táctico
Función de derivación	$An = \frac{P_{DenCC}(DenCC) + P_{DenCR}(DenCR) + P_{DenCom}(DenCom) + P_{DenDefAn}(DenDefAn)}{P_{DenCC} + P_{DenC} + P_{DenCom} + P_{DenDefAn}}$
Atributos	**An:** Capacidad de ser analizado del producto **P_{DenCC} y DenCC:** Peso y nivel de calidad de la densidad de la complejidad ciclomática **P_{DenCR} y DenCR:** Peso y nivel de calidad de código repetido **P_{DenCom} y DenCom:** Peso y nivel de calidad de la densidad de comentarios **$P_{DenDefAn}$ y DenDefAn:** Peso y nivel de calidad del índice de capacidad de ser analizado

Tabla 5. Capacidad de ser analizado.

Finalmente, una vez definidas las funciones para el cálculo de los indicadores de las subcaracterísticas en las que se divide la mantenibilidad, en KEMIS se define otra función para obtener el indicador de mantenibilidad, que, al igual que ocurre con el cálculo de subcaracterísticas, hace uso de pesos, dando más importancia a unas subcaracterísticas que a otras. En la Tabla 6 se muestra esta función.

Mantenibilidad	
Nivel	Estratégico
Función de derivación	$Ma = P_{An}(An) \times P_{Cam}(Cam) \times P_{Est}(Est) \times P_{Pru}(Pru)$
Atributos	**Ma:** Mantenibilidad del producto **P_{An} y An:** Peso y nivel de calidad de la capacidad de ser analizado **P_{Cam} y Cam:** Peso y nivel de calidad de la capacidad de cambios **P_{Est} y Est:** Peso y nivel de calidad de la estabilidad **P_{Pru} y Pru:** Peso y nivel de calidad de la capacidad de pruebas

Tabla 6. Mantenibilidad.

4.4. Modelos de calidad

Con todos los elementos aportados por KEMIS se genera un modelo de calidad que se almacena en una base de datos, que permite configurar los pesos de las funciones expuestas, los umbrales y las relaciones entre atributos y subcaracterísticas. Así es posible la evolución del modelo de calidad que se empleará en una organización.

A la hora de implantar KEMIS en una empresa, se debe favorecer la modificación del modelo de calidad inicial, por necesidades o restricciones propias de la empresa o, simplemente, por la evolución natural del modelo de calidad.

KEMIS permite modificar valores y relaciones de diferentes elementos del modelo de calidad: cambios en pesos, cambios en umbrales e inclusión o no de un atributo (por ejemplo, la Capacidad para ser probado) en el modelo de calidad que se va a usar, etc. En cualquier caso, un cambio en el modelo de calidad supone un cambio en los criterios de medición y, por tanto, los resultados que se obtengan seguramente serán diferentes.

Para solucionar este problema, KEMIS permite el versionado de Modelos de Calidad, de tal forma que se proporciona un mecanismo para fijar los criterios de medición (una versión concreta del modelo de calidad) con el resultado de la medición, a la vez que es posible la evolución del modelo de calidad.

Además, KEMIS posibilita la existencia conjunta de diferentes modelos de calidad, lo que permite la medición de un sistema con diferentes versiones de estos modelos. Así no sólo se facilita la evolución de los modelos de calidad, sino también su comparación.

4.5. Informes y representación

Una vez obtenida una medición basada en el modelo de calidad propuesto por la ISO/IEC 9126, se pueden obtener representaciones gráficas de diagramas de barras o incluso diagramas de Kiviat, como el de la Figura 3.

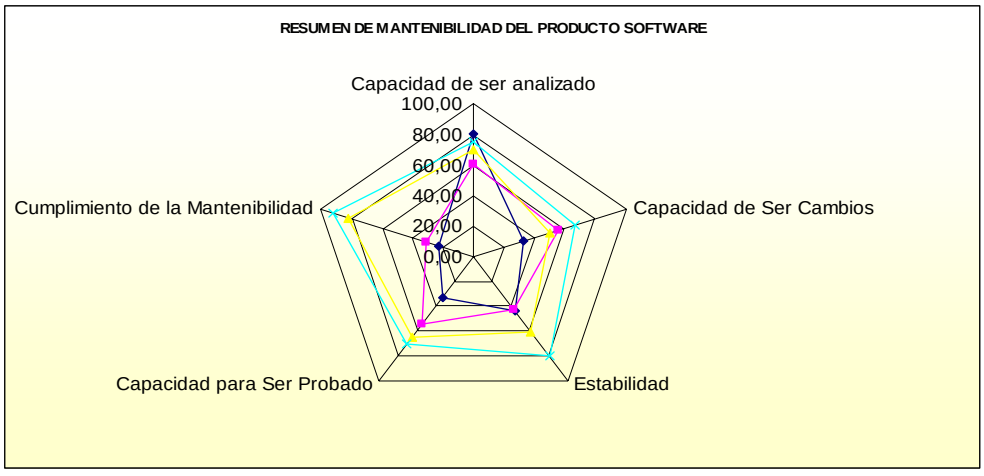


Figura 3. Kiviat de la característica de mantenibilidad de un producto.

5. Un ejemplo de evaluación

La Figura 4 muestra un ejemplo de evaluación de la calidad de varios productos de software libre disponibles en la web. Se han obtenido tres de los valores que componen la mantenibilidad: Estabilidad, Capacidad para ser cambiado y Capacidad para ser analizado. De esta manera, podemos apreciar el nivel de mantenibilidad. La columna NCSS corresponde al número de líneas de código no comentadas de cada producto.

				Subcaracterísticas		
				Mantenibility		
NOMBRE	PROVEEDOR	NIVEL mantenibilidad DE PRODUCTO	NCSS	Stability	Changeability	Analysability
jboss-maven-plugin	CodeHaus	83,05	371,00	87,26	100,00	72,47
JDepend	Clarkware Consulting	75,35	1.808,00	100,00	100,00	50,70
castor-maven-plugin	CodeHaus	72,10	263,00	98,38	100,00	45,00
jlgui	javazoom	71,81	8.538,00	85,65	50,00	75,80
quilt	apache	60,91	4.087,00	95,18	49,58	49,43
ivy	apache	60,07	24.612,00	100,00	50,00	45,13
commons-collections	apache	60,00	19.417,00	100,00	50,00	45,00
FindBugs	University of Mariland	59,85	72.195,00	99,40	50,00	45,00
JavaNCSS	Klcee	59,42	15.203,00	100,00	50,00	43,84
TightVNCviewer	TightVNC Software	56,30	3.292,00	3,80	100,00	60,70
velocity-tools	apache	52,99	4.960,00	21,98	100,00	45,00
CheckStyle	Oliver Burn	39,46	28.744,00	17,85	50,00	45,00
PMD	InfoEther	37,65	39.604,00	10,60	50,00	45,00
commons-logging	apache	34,67	1.774,00	0,00	48,69	45,00

Figura 4. Ejemplo de evaluación.

6. Conclusiones

Las normas ISO 9126 y 25000 establecen criterios para la especificación de requisitos de calidad de los productos de software, sus métricas y su evaluación.

En esta línea se desarrolla KEMIS, permitiendo la obtención de atributos de calidad y el escalado de información hacia niveles tácticos y estratégicos, basándose en un estándar, el ISO/IEC 9126. En lo que respecta a trabajos futuros, en la actualidad se están desarrollando ampliaciones del modelo que se presenta para cubrir más áreas de la ISO 25000 (funcionalidad, seguridad, etc.), destacando las pruebas de software o técnicas de caja negra. Otra línea de trabajo es la explotación de los datos recogidos por la infraestructura tecnológica, aplicando técnicas de control estadístico.

Agradecimientos

Este trabajo se ha desarrollado dentro del proyecto KEMIS, el cual es financiado por la empresa Kybele Consulting.

Referencias

[1] Garvin, D., "What is Quality?", *Sloan Management Review*, otoño de 1984.

[2] Humphrey, W., *Managing the Software Process*, Addison-Wesley, 1989.

[3] INTECO, *Estudio sobre la certificación de la calidad como medio para impulsar la industria de desarrollo del software en España*, INTECO, 2008.

[4] SEI, *Standard CMMI Appraisal Method for Process Improvement (SCAMPISM) v1.1/v1.2 Class A Appraisals Using Capability Maturity Model Integration (CMMI) v1.1 /v1.2 *2007 Year-End Update*, Software Engineering Institute, 2008.

[5] Kitchenham, B. y Pfleeger, S. L., "Software Quality: The Elusive Target", *IEEE Software*, vol. 20, nº 1, pp. 12-21, 1996.

[6] Maiabaum, T.. y Wassyng, A., "A Product-Focused Approach to Software Certification", Computer, vol. 41, num. 2, pp. 91-93, 2008.

[7] ISO (International Organization for Standardization), *ISO/IEC 25022 Software Engineering. Software Product Quality Requirements and Evaluation (SQuaRE). Measurement of Internal Quality*, ISO, 2005.

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

[9] ISO (International Organization for Standardization), *ISO 14598: 1999-2001. Information Technology. Software Product Evaluation. Parts 1-6*, ISO, 1999.

[10] Fenton, N. y Pfleeger S., *Software Metrics: A Rigorous Approach*, Chapman & Hall, 1997.

[11] McCabe, T., "A Software Complexity Measure", *IEEE Transactions on Software Engineering*, vol. 2, num. 4, pp. 308-320, 1976.

[12] Hitz, M.y Montazeri, B., "Chidamber and Kemerer's metrics suite: A Measurement Theory Perspective", *IEEE Transactions on Software Engineering*, vol. 22, num. 4, pp.267-271, 1996.

[13] Brito e Abreu, F. y Carapuça, R., "Object-Oriented Software Engineering: Measuring and Controlling the Development Process", *4th Int Conf. on Software Quality. USA*, 1994.

[14] Erlikh, L., "Leveraging Legacy System Dollars for E-business", *IT Professional*, mayo/junio, vol. 2, nº 3, pp. 17-23, 2000.

Revista Española de Innovación, Calidad e Ingeniería del Software, Vol.4, No. 2, 2008

Modelo de calidad para herramientas FLOSS que dan apoyo al modelado de procesos del negocio

Leslibeth Pessagno, Kenyer Domínguez, Lornel Rivas,
María Pérez, Luis E. Mendoza, Edumilis Méndez

Laboratorio de Investigación en Sistemas de Información (LISI).
Departamento de Procesos y Sistemas. Universidad Simón Bolívar
leslibethpessagno@gmail.com
{kdoming, lrivas, movalles, lmendoza, emendez}@usb.ve

Abstract

The selection of FLOSS tools supporting business modeling discipline is a complicated task; besides verifying the proper use of language and notations such as BPMN [1], UML [2] and SPEM, we must validate that such tools meet the features of this type of software. Consequently, the quality of these tools should be assessed through a quality model that allows for determining fulfillment of such requirements. This research, currently in progress, included the use and creation of a Quality Systemic Model (MOSCA) [3]. This model is based on ISO 9126 [4], Dromey's [5] quality model, and the Goal/Question/Metric paradigm [6]. In addition, the instantiation proposed includes software attributes relating to functionality, usability, and maintainability and establishes 75 new metrics, for a total of 128 metrics, to assessing FLOSS tools for Business Modeling purposes. Four tools were selected in order to validate this proposal, namely Eclipse Process Framework Composer (EPFC) [7], StarUML [8], Intalio [9] and y DIA (http://www.gnome.org/projects/dia/), the chosen tool being EPFC, the highest quality tool to support this discipline.

Key words: business modeling, software engineering tools, FLOSS, ISO 9126, software quality model.

Resumen

Seleccionar herramientas FLOSS que dan apoyo a la disciplina modelado del negocio es una tarea complicada ya que, además de verificar la correcta utilización de lenguajes y notaciones como BPMN [1], UML [2] y SPEM, se debe validar que satisfacen las propiedades que caracterizan a este tipo de software. Por tanto, es necesario evaluar la calidad de estas herramientas a través de un modelo de calidad que permita determinar el cumplimiento de dichos requisitos. En esta investigación en progreso se utilizó y se realizó un instanciación del modelo sistémico de calidad (MOSCA) [3]. Este modelo se basa en los estándares de ISO 9126 [4], en el modelo de calidad de Dromey [5] y en el paradigma de objetivos, preguntas y métricas [6]. La instanciación propuesta incluye atributos del software relacionados con la funcionalidad, la usabilidad y la mantenibilidad, estableciendo 75 nuevas métricas para un total de 128 que permiten evaluar herramientas FLOSS para el modelado del negocio. Con el fin de validar esta propuesta se seleccionaron cuatro

herramientas: Eclipse Process Framework Composer (EPFC) [7], StarUML [8], Intalio [9] y DIA (http://www.gnome.org/projects/dia/). EPFC se escogió como la herramienta con más alto nivel de calidad en cuanto a apoyo a esta disciplina.

Palabras clave: modelado del negocio, herramientas de ingeniería del software, FLOSS, ISO 9126, modelo de calidad de software.

1. Introducción

La versatilidad de las herramientas de ingeniería del software que dan apoyo al modelado del negocio basadas en software libre, su amplia variedad en el mercado y el diverso nivel de funcionalidad que ofrecen, hacen compleja su selección. Además, existen distintos lenguajes de modelado que permiten representar procesos del negocio —entre ellos, EPM [10] y BPMN [1]—; incluso existen algunos especializados solo en procesos de desarrollo de software, como el perfil de negocios de UML [2], SPEM y la extensión de Eriksson y Penker [11], lo cual hace aún más complicada la evaluación de este tipo de herramientas.

En el mercado existen herramientas que dan apoyo a los lenguajes mencionados antes. Algunas de ellas se caracterizan por combinar las características de software libre [12] y Software Open Source [13], que se pueden resumir en los siguientes aspectos: acceso al código fuente, modificación del código, sin restricciones de uso, copia y re-distribución [12, 15, 16]. No obstante, seleccionar este tipo de herramientas no es una tarea fácil, pues deben satisfacer, además de los requisitos de modelado de procesos del negocio, las propiedades de Free/Libre Open Source Software (FLOSS) [14].

Por esta razón, la presente investigación pretende alcanzar los siguientes objetivos: (a) propuesta de una instanciación del modelo sistémico de calidad (MOSCA) [3] que permita evaluar la calidad de herramientas FLOSS que den apoyo al modelado del negocio; (b) evaluación de la utilidad del modelo aplicándolo a la estimación de la calidad de un conjunto de herramientas con estas características.

En la bibliografía existente se han encontrado dos trabajos relacionados con el que se presenta en este artículo. Por un lado está el método de cualificación y selección de Software Open Source [15] que, debido a sus características, requiere considerar criterios asociados a la modificación, licencias y madurez de software, entre otros. Y por otro lado, existe un meta-modelo para evaluar lenguajes de modelado de procesos del negocio [16] que permite representar un amplio rango de conceptos de procesos del negocio. Para

garantizar su cobertura, el meta-modelo está conformado por cinco perspectivas: funcional, organizativa, comportamiento, información y contexto de procesos del negocio. Sin embargo, ninguna de estas dos iniciativas está orientada a establecer un modelo sistémico de calidad para herramientas FLOSS que dan apoyo al modelado de procesos del negocio.

Este trabajo consta de siete secciones. En esta se presenta la introducción y seguidamente, la metodología empleada en la investigación. El epígrafe tercero describe MOSCA; el cuarto presenta el modelo de calidad propuesto para herramientas FLOSS que dan apoyo al modelado del negocio. Las secciones quinta y sexta describen la aplicación del modelo y los resultados obtenidos. Por último, en la sección séptima se presentan las conclusiones y recomendaciones.

2. Metodología

En este trabajo se utilizó el Framework Metodológico Sistémico para investigar sistemas de información [17], el cual se basa en el método de investigación-acción [18] y en la metodología DESMET [19]. El primero se desarrolla en cinco fases: diagnosticar, planificar la acción, realizar la acción, evaluar y especificar el aprendizaje [17], mientras que la metodología DESMET interviene para complementar la evaluación del modelo. Propone nueve métodos de evaluación, de los cuales se empleó el método de análisis de las características por estudio de caso [17]. Además, se incluyó el enfoque basado en objetivos, preguntas y métricas, con el propósito de hacer posible la medición del software en un contexto de mejora de la calidad [6].

3. Modelo de calidad sistémico (MOSCA)

El modelo sistémico de calidad se basa en la matriz de calidad global sistémica de Callaos e integra tres modelos de calidad [20, 21]: producto, proceso de desarrollo y perspectiva humana. MOSCA está constituido por estos niveles:

- Nivel 0. Dimensiones. Aspectos internos y contextuales del producto, el proceso y la perspectiva humana.
- Nivel 1. Categorías. Se contemplan 14 categorías, cinco pertenecientes al proceso, seis pertenecientes al producto y tres para la perspectiva humana.
- Nivel 2. Características. Cada categoría tiene asociado un conjunto de

características que definen las áreas claves para lograr, controlar y asegurar la calidad en las tres perspectivas. Hay 56 características asociadas para el producto, 27 para la perspectiva del proceso y 15 para la parte humana.

- Nivel 3. Métricas, usadas para medir la calidad sistémica. Existen 715 métricas.

MOSCA evalúa el producto según normas internacionales, pues las categorías presentadas antes coinciden con las características del estándar ISO 9126 [4], que son establecidas para garantizar la calidad de producto de software.

Para la aplicación de MOSCA existe el siguiente algoritmo:

1. Estimación de la calidad del producto. Inicialmente, se debe medir la categoría de funcionalidad del producto. Si esta cumple con el 75% de las características necesarias propuestas para esta categoría, se prosigue con el siguiente paso.
2. Instanciación del submodelo del producto. De las cinco categorías restantes, se seleccionan dos. El algoritmo recomienda trabajar con un máximo de tres categorías, pues si se seleccionan más, podrían presentar conflictos.
3. Estimación de la calidad para cada categoría. Para las dos categorías seleccionadas en el paso anterior se debe: (a) aplicar las métricas propuestas en el submodelo del producto para las categorías seleccionadas; (b) verificar que el 75% de las métricas están dentro de los valores óptimos (mayor o igual a tres) para cada una de sus características; y (c) evaluar la categoría. Para que una categoría sea satisfecha, al menos el 75% de sus características deben ser altamente satisfechas. Esto garantiza coherencia y consistencia con los niveles de aceptación establecidos por el modelo.
4. Estimación de la calidad del producto partiendo de las categorías evaluadas. Si no se satisface la categoría funcionalidad, el algoritmo finaliza y la calidad del producto de software será nula. Si un producto cumple con los objetivos para los cuales se creó (funcionalidad), tendrá una calidad básica. Si satisface solo una de las categorías seleccionadas, además de la funcionalidad, tendrá un nivel de calidad intermedio; si satisface todas las categorías seleccionadas, tendrá un nivel avanzado.

4. Modelo de calidad propuesto para herramientas FLOSS que dan apoyo al modelado de procesos del negocio

De acuerdo con el algoritmo de MOSCA, para el proceso de evaluación se deben elegir tres

de las seis categorías de la dimensión producto, entre las cuales ha de incluirse la funcionalidad. Las otras dos categorías seleccionadas son la usabilidad (pues las herramientas de modelado deben ser fáciles de comprender y utilizar) y la mantenibilidad. Dado que el alcance de esta investigación cubre herramientas FLOSS, además estas categorías también son propuestas en [15] entre sus criterios de evaluación. Para cada una de estas categorías se seleccionó un subconjunto de características y para estas, un subconjunto de métricas; en algunos casos fue necesario añadir nuevas métricas. A continuación se expone en qué consiste cada una de estas categorías, así como los criterios utilizados para seleccionar sus características.

4.1. Funcionalidad

La funcionalidad es la capacidad del producto de software para proveer funciones que cumplan con necesidades específicas o implícitas cuando se usa en unas condiciones determinadas [4]. Es una categoría fundamental porque se espera que todo producto cumpla con los propósitos para los que fue creado; en este caso, es necesario que la herramienta que se va a evaluar dé apoyo, al menos, a los lenguajes UML, SPEM, BPMN y EPM. De las ocho características de esta categoría se escogieron las descritas en la Tabla 1.

Categoría	**Definición y justificación**
Ajuste a los propósitos (FUN 1)	Es la capacidad del producto de software para proveer un conjunto de funciones apropiado según tareas y objetivos específicos del usuario. En este trabajo se requiere que una herramienta cuente con funcionalidades que permitan representar modelos de procesos del negocio.
Precisión (FUN 2)	Es la capacidad del producto de software para proveer los resultados correctos. La herramienta que dé apoyo a lenguajes para modelado del negocio debe proveer la notación adecuada para la representación de los procesos que lo conforman.
Interoperatividad (FUN 3)	Es la capacidad del producto de software para interactuar con uno o más sistemas especificados. Dado que la herramienta permitirá representar modelos de procesos del negocio y que estos pueden estar compuestos por uno o más de ellos, hay que saber si el producto tiene funcionalidades utilizadas por otro sistema o si otros sistemas utilizan sus funcionalidades para llevar a cabo el modelado.
Corrección (FUN 5)	Esta característica se divide en tres categorías relacionadas con la capacidad de cómputo, completitud y consistencia. Alguna violación de una de estas propiedades puede significar que el software no tenga la funcionalidad esperada. Es necesario que los diagramas generados por la herramienta sean completos y consistentes.
Encapsulado (FUN 7)	Las variables, las constantes y los tipos se deben usar en el contexto en el que son definidos. El modo en que se usen puede tener un impacto significativo sobre la modularidad y, por lo tanto, sobre la calidad de los módulos y programas.

Tabla 1. Características para la categoría funcionalidad.

A fin de de cumplir con los requisitos necesarios para representar modelos de procesos de negocio, se añadieron las subcaracterísticas presentadas en la Tabla 2.

Subcaracterística	Descripción
Diagramas (FUN 1.1)	Es la capacidad de la herramienta para llevar a cabo la representación de modelos de procesos de negocio a través de los diagramas asociados a un lenguaje para el modelado del negocio, como BPMN, EPM, Extensión Eriksson & Penker, SPEM o el perfil UML para negocios.
Documentación (FUN 1.2)	Es la capacidad de la herramienta de proveer mecanismos para la generación de documentación, así como para permitir realizar anotaciones en los diagramas.
Lenguajes (FUN 1.3)	Se refiere a los lenguajes empleados para representar modelos de procesos de negocio que la herramienta es capaz de proveer al usuario.
Detalles de abstracción (FUN 2.1)	Es la capacidad de la herramienta de modelado para detallar las abstracciones que representan un modelo de procesos del negocio.
Completo (FUN 5.1)	Es la capacidad de la herramienta de proveer todos los estereotipos, conectores, símbolos, etc., pertenecientes al lenguaje de modelado. También permite verificar si esta posee todos los diagramas asociados al lenguaje.
Consistente (FUN 5.2)	Es la capacidad de verificar si los símbolos provistos por la herramienta corresponden a los del lenguaje de modelado al cual pertenecen y si estos no pueden ser mezclados o intercambiados entre lenguajes.
Taxonomía (FUN 7.1)	Es la capacidad de organizar los diagramas y lenguajes a través de alguna clasificación.

Tabla 2. Subcaracterísticas propuestas para la categoría funcionalidad de herramientas FLOSS que dan apoyo al modelado del negocio.

4.2. Usabilidad

La usabilidad es la capacidad del producto de software para ser atractivo, entendido, aprendido y utilizado por el usuario en unas condiciones específicas [4]. Las herramientas de modelado deben permitir realizar la diagramación de forma sencilla, y han de presentar las funcionalidades de forma accesible y consistente, para que los distintos usuarios que las usen trabajen cómodamente, reduciendo el tiempo y esfuerzo de aprendizaje [22]. Por ello, de las 11 características de esta categoría se seleccionaron las cuatro descritas en la Tabla 3.

Categoría	Definición y justificación
Facilidad de comprensión (USA 1)	Es la capacidad del producto de software para facilitar al usuario entender el software y la forma en que se puede usar para efectuar diferentes tareas bajo condiciones específicas. Es fundamental que las funciones asociadas a la diagramación sean fáciles de comprender y ubicar por los diferentes tipos de usuarios (programadores, analistas, etc.).
Interfaz gráfica (USA 3)	Se refiere a la capacidad del producto de software para captar la atención del usuario. La interfaz gráfica debe agradar visualmente al usuario, además de presentar consistentemente la información y las acciones de que dispone la herramienta.
Operatividad (USA 4)	Es la capacidad del producto de software para facilitar al usuario su uso y control. Es importante que la herramienta requiera el mínimo esfuerzo para su utilización, pues el proceso de diagramación debe sencillo y rápido de realizar.
Autodescriptivo (USA 11)	Una forma estructural es autodescriptiva si su propósito es evidente en el nombre de los módulos y los identificadores tienen significado en el contexto de la aplicación. Es necesario considerar esta característica, pues si la herramienta es autodescriptiva, se podrá operar fácilmente con ella.

Tabla 3. Características para la categoría usabilidad.

La Tabla 4 muestra las subcaracterísticas añadidas al modelo, así como su descripción.

Característica	**Subcaracterística**	**Descripción**
Facilidad de comprensión (USA 1)	Ergonomía (USA 1.1)	Es la capacidad que tiene la interfaz de la herramienta para facilitar la interacción hombre-máquina.
Operatividad (USA 4)	Control de errores (USA 4.1)	Se refiere a los mecanismos que provee la herramienta para que el usuario pueda recuperarse de errores.
	Documentación (USA 4.2)	Se trata de la documentación de las funcionalidades provistas por la herramienta.

Tabla 4. Subcaracterísticas y métricas propuestas para la categoría usabilidad para las herramientas FLOSS que dan apoyo al modelado del negocio.

4.3. Mantenibilidad

Mantenibilidad es la capacidad del producto de software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software ante cambios del ambiente, en requisitos y especificaciones funcionales [4]. Las herramientas desarrolladas en FLOSS, a diferencia de las de software propietario, deben prestar especial atención a aspectos relacionados con la visibilidad, reutilización y modificación. Por esta razón, de las catorce características asociadas a esta categoría, se seleccionaron las descritas en la Tabla 5.

Categoría	**Definición y justificación**
Capacidad de análisis (MAB 1)	Es la capacidad del producto de software para ser diagnosticado por deficiencias o causas de fallos en el software, o por partes que hay que modificar. Para poder evaluar la herramienta y determinar cuáles serán las mejoras que pueden llevarse a cabo, es necesario que esta permita realizar fácilmente el diagnóstico.
Capacidad de cambio (MAB 2)	Es la capacidad del producto de software para facilitar la implementación de una modificación específica. Es conveniente que una herramienta FLOSS posibilite a la comunidad de software libre a realizar aportes que permitan incurrir en potenciales mejoras.
Estabilidad (MAB 3)	Es la capacidad del producto de software para evitar efectos inesperados después de modificaciones en el software. Se requiere que, al modificar la herramienta, no se generen efectos colaterales en funcionalidades que trabajaban correctamente.
Acoplamiento (MAB 5)	Es una medida de la interconexión entre los módulos de una estructura de programa. En el diseño de software se intenta conseguir el menor nivel posible de acoplamiento, ya que facilita la modificación del software. Ocurre todo lo contrario cuando el diseño presenta un acoplamiento fuerte.
Cohesión (MAB 6)	Una forma estructural es cohesiva si todos sus elementos están enlazados estrechamente unos a otros y si contribuyen a llevar a cabo un simple objetivo o función. Una alta cohesión en herramientas FLOSS facilitaría su reutilización y evitaría la aparición de efectos colaterales durante la modificación del software.
Atributos de madurez del software (MAB 8)	Son el conjunto de las características físicas y medidas asociadas a la edad y uso del sistema de software objetivo. Una herramienta debe ser lo suficientemente madura para aumentar las posibilidades de éxito en las modificaciones realizadas.

Tabla 5. Características para la categoría mantenibilidad.

La Tabla 6 presenta las subcaracterísticas que se añadieron a la categoría

mantenibilidad.

Característica	**Subcaracterística**	**Descripción**
Capacidad de análisis (MAB 1)	Legibilidad del código (MAB 1.1)	Se refiere a la capacidad del código fuente de ser leído por cualquier desarrollador, incluso por aquel que no pertenezca al proyecto.
	Licencia (MAB 2.1)	Son las propiedades que tiene la licencia de la herramienta. Entre otras cosas, la licencia debe garantizar las libertades del usuario para acceder al código fuente, modificarlo, copiarlo o distribuirlo
Capacidad de cambio (MAB 2)	Modificación (MAB 2.2)	Es la capacidad que tiene el código fuente de la herramienta para ser modificado.
	Documentación (MAB 2.3)	Se refiere a la documentación técnica del software: documentación del código fuente, de diseño, plan de riesgos, etc.
Estabilidad (MAB 3)	Servicios (MAB 3.1)	Se refiere a los servicios de soporte, entrenamiento y consultoría provistos por la comunidad de desarrollo de la herramienta para respaldar a los usuarios.
Atributos de madurez (MAB 8)	Adopción (MAB 8.1)	Permite medir el grado de aceptación de la herramienta en el mercado, tanto por personas física, como por empresas y organizaciones.

Tabla 6. Subcaracterísticas propuestas para la categoría mantenibilidad para herramientas FLOSS que dan apoyo al modelado del negocio.

La instanciación de MOSCA para herramientas FLOSS para el modelado del negocio presenta 128 métricas, de las cuales 75 son métricas nuevas (42 en funcionalidad, nueve en usabilidad y 24 en mantenibilidad). La Figura 1 muestra el subárbol de la adaptación.

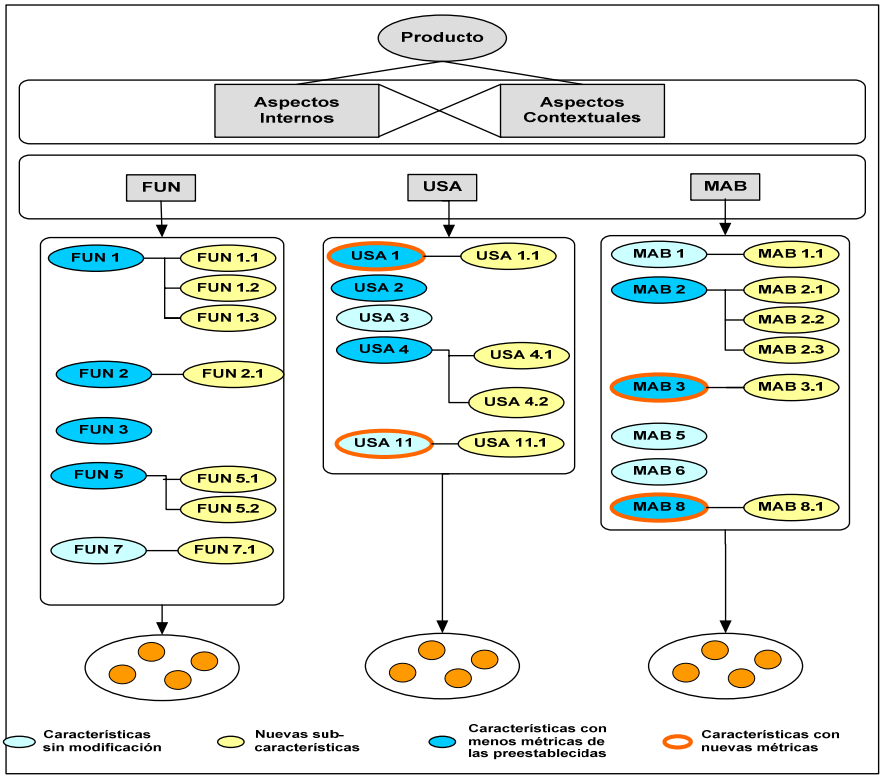


Figura 1. Instanciación de MOSCA para herramientas FLOSS que dan apoyo al modelado del negocio.

5. Aplicación de la instanciación

La instanciación de MOSCA se aplicó a cuatro herramientas: Eclipse Process Framework Composer (EPFC) [7], StarUML [8], Intalio Designer [9] y DIA. Todas las herramientas permiten modelar procesos del negocio utilizando lenguajes como BPMN [1], UML (perfil de negocios [2]), SPEM y EPM [10]. EPFC, además de ser una herramienta empleada para representar diagramas, permite gestionar procesos. Sin embargo, para este trabajo de investigación es de interés estudiar un subconjunto de la parte funcional, específicamente, el modelado visual de procesos del negocio. Los resultados obtenidos en la evaluación se detallan en la Tabla 7.

En total se invirtieron aproximadamente 88 horas para la aplicación de la instanciación. El proceso se realizó de la siguiente forma: se descargaron del site de la comunidad de desarrollo correspondiente las últimas versiones de las herramientas y luego se instalaron (una hora por herramienta). Posteriormente, se ejecutaron las aplicaciones una a una para iniciar el proceso de familiarización, cuyo tiempo estimado fue de cuatro horas por herramienta; después se verificó el cumplimiento o no de las métricas de funcionalidad (tres horas por herramienta) y usabilidad (una hora por herramienta). Para la evaluación de las métricas de mantenibilidad se revisó la información y documentación existente en los sites oficiales de las herramientas (siete horas por herramienta).

Categoría	Característica	Subcaracterística	EPFC	StarUML	Intalio	DIA
Funcionalidad	Ajuste a los propósitos	Diagramas	100,00	82.35	100,00	55,56
		Documentación	100,00	100,00	60,00	60,00
		Lenguajes	50,00	75,00	60,00	55,00
	Precisión	Detalles de abstracción	100,00	70,00	60,00	50,00
	Interoperatividad	Interoperatividad	46,67	20,00	20,00	20,00
	Corrección	Completo	76,36	93,84	100,00	93,33
		Consistente	60,00	66,67	100,00	46,67
	Encapsulado	Taxonomía	100,00	46,67	-	-
		Porcentaje de satisfacción	77,78	78,69	77,14	64,67
Usabilidad	Facilidad de comprensión	Facilidad de comprensión	64,00	80,00	84,00	92,00
		Ergonomía	100,00	100,00	100,00	100,00
	Capacidad de aprendizaje	Capacidad de aprendizaje	80,00	80,00	80,00	20,00
	Interfaz gráfica	Interfaz gráfica	100,00	94,29	94,29	65,71
	Operatividad	Operatividad	86,00	72,00	52,00	64,00
		Control de errores	100,00	100,00	100,00	100,00
		Documentación	80,00	96,00	64,00	84,00
	Autodescripción	Autodescripción	100,00	100,00	90,00	100,00
		Porcentaje de satisfacción	86,25	85,63	74,38	75,00

Mantenibilidad	Capacidad de análisis	Capacidad de análisis	100,00	80,00	30,00	100,00
		Legibilidad del código	100,00	100,00	20,00	100,00
	Capacidad de cambio	Capacidad de cambio	100,00	68,00	28,00	72,00
		Licencia	100,00	100,00	77,14	100,00
		Modificación	100,00	53,33	33,33	80,00
		Documentación	100,00	60,00	20,00	20,00
	Estabilidad	Estabilidad	73,33	20,00	20,00	20,00
		Servicios	100,00	20,00	100,00	60,00
	Acoplamiento	Acoplamiento	100,00	100,00	20,00	60,00
	Cohesión	Cohesión	100,00	100,00	20,00	80,00
	Atributos de madurez del software	Atributos de madurez del software	84,44	66,67	44,44	75,56
		Adopción	84,00	68,00	20,00	36,00
Porcentaje de satisfacción			93,48	70,43	40,00	70,00
Nivel de calidad			Avanzada	Media	Básica	Nula

Tabla 7. Resultados de la evaluación.

6. Resultado del análisis

La Figura 2 muestra los resultados obtenidos en cada categoría. En funcionalidad, EPFC obtuvo un 77,78%; StarUML, un 78,69%; Intalio, un 77,14% y DIA, un 64,67% de métricas satisfechas. De acuerdo con el algoritmo de MOSCA, DIA no alcanzó el nivel mínimo de aceptación requerido, que es el 75%. También se observa que, en cuanto a los resultados asociados a la categoría usabilidad, EPFC obtuvo la puntuación más alta, con el 86,25%; el segundo y el tercer lugar lo ocupan StarUML y DIA, respectivamente. Intalio no alcanzó el porcentaje mínimo requerido. Finalmente, en la categoría mantenibilidad, la única herramienta que con una puntuación mayor al 75% es EPFC, con un 93,48%.

DIA no satisfizo el porcentaje mínimo requerido en la funcionalidad. Esto la califica como una herramienta con un nivel de calidad nulo. Por su parte, Intalio obtuvo una puntuación mayor al 75% únicamente en la categoría de funcionalidad, por lo que tiene un nivel de calidad básico. StarUML satisface dos de sus categorías: funcionalidad y usabilidad, por lo que posee un nivel de calidad intermedio.

Finalmente, la única herramienta que satisface las tres categorías seleccionadas para la instanciación, con una puntuación superior al 75% es EPFC. En consecuencia, presenta un nivel de calidad avanzado. Por esta razón, EPFC es la herramienta seleccionada para realizar las mejoras relacionadas con las características que sean pertinentes y factibles para la investigación. La presentación de estos resultados escapa del alcance de este artículo y

serán publicados en el futuro.

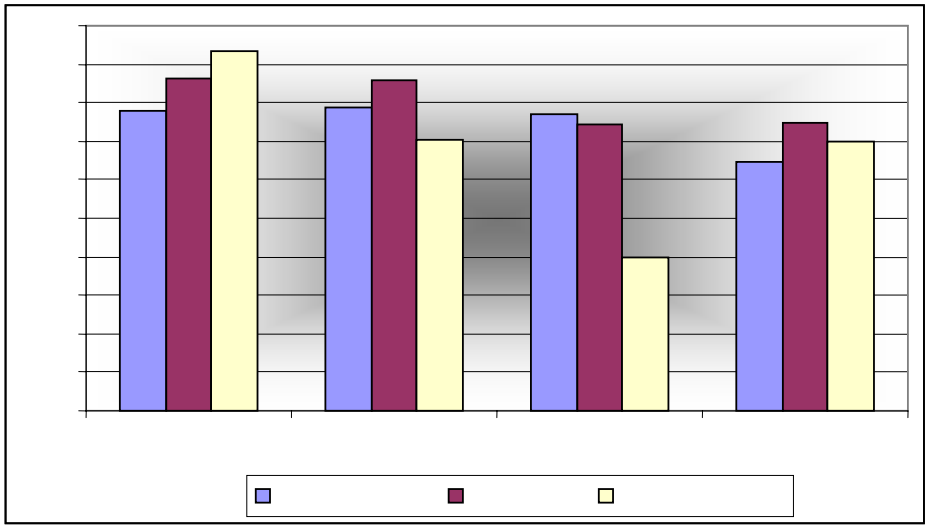


Figura 2. Resultados de las categorías funcionalidad, usabilidad y mantenibilidad para cada herramienta.

7. Conclusiones y recomendaciones

Este trabajo propone una instanciación de MOSCA que permite evaluar la calidad de herramientas de ingeniería del software basadas en FLOSS, que den apoyo al modelado del negocio, las cuales, además de cubrir las funcionalidades requeridas, deben ser fáciles de utilizar y ser adaptables al cambio. Este modelo fue aplicado a EPFC [7], StarUML [8], Intalio [9] y DIA, con la finalidad de comprobar la utilidad del modelo y, además, seleccionar la herramienta con un nivel de calidad avanzado para mejorarla en un futuro. En este caso, la seleccionada fue EPFC. Los resultados de esta investigación pueden servir de guía para pequeñas y medianas empresas que precisen seleccionar una herramienta de apoyo a esta disciplina. Siendo EPCF una herramienta FLOSS, puede resultar de fácil acceso para estas organizaciones. Finalmente, se debe indicar que MOSCA puede ser adaptado para herramientas de modelado del negocio con características específicas. Por esta razón, para futuros trabajos de investigación se recomienda evaluar herramientas que den apoyo a la gestión de procesos del negocio.

Agradecimientos

Esta investigación ha sido financiada por el Fondo Nacional de Ciencia, Tecnología e Innovación (FONACIT) de la República Bolivariana de Venezuela, a través del proyecto G-2005000165.

Referencias

[1] Business Process Management Initiative (BPMI), *Business Process Modeling Notation Specification BPMN 1.0*, OMG, 2006.

[2] Johnston, S., *Rational UML Profile for Business Process*, IBM, 2004.

[3] Mendoza, L., Pérez, M. y Grimán, A., "Prototipo de modelo sistémico de calidad (MOSCA) del software", *Computación y Sistemas*, vol. 8, nº 3, pp. 196-217, 2005.

[4] ISO, *ISO/IEC 9126-1, Software Engineering. Product Quality. Part 1: Quality Model*, ISO, 2001.

[5] Dromey, G., "A Model for Software Product Quality", *IEEE Transactions on Software Engineering*, vol. 21, nº 2, pp. 146-162, 1995.

[6] Basili, V., Caldiera, G. y Rombach, H., "The Goal Question Metric Approach", en: Marciniak, J. J. (ed.), *Encyclopedia of Software Engineering*, Wiley, pp. 528–532, 2001.

[7] Eclipse Process Framework Project (EPF), *Documentación del proyecto*, EPF, 2006. http://www.eclipse.org/epf/ (consultado en marzo de 2008)

[8] StarUML, *UML2.0 Modeling Tool. Multilingual Project. Version 5.0.2.1570*, StarUML, 2005. http://staruml.sourceforge.net/en/ (consultado en abril de 2008)

[9] Intalio, *Intalio Designer, herramienta de modelado con notación para modelado de procesos del negocio*. http://www.intalio.com/ (consultado en mayo de 2008)

[10] Stemberger, M., Jaklic, J. y Popovic, A., "Suitability of Process Maps for Business Process Simulation in Business Process Renovation Projects", *16TH EUROPEAN SIMULATION Symposium ESS2004. Budapest (Hungary)*, pp. 109-123, 2004.

[11] Eriksson, H. y Penker, M., *Business Modeling with UML. Business Patterns at Work*, Wiley, 2000.

[12] Free Software Foundation, *The Free Software Definition*, 2007. http://www.fsf.org/licensing/essays/free-sw.html (consultado en enero de 2008)

[13] Open Source Initiative, *The Open Source Definition*, 2007. http://opensource.org/docs/osd (consultado en enero de 2008)

[14] FLOSS, *Free/Libre and Open Source Software: Survey and Study*. http://www.flossproject.org/outline.htm (consultado en enero de 2008)

[15] QSOS Method, *QSOS Introduction*, 2006. http://www.qsos.org/?page_id=7 (consultado en abril de 2008)

[16] List, E. y Korherr, B., "An Evaluation of Conceptual Business Process Modelling", en: Haddad, H. M. (ed.), *Languages Symposium on Applied Computing. Proceedings of the 2006 ACM Symposium on Applied Computing*, pp. 1.532-1.539, 2006.

[17] Pérez, M., Grimán, A., Mendoza, L. y Rojas, T., "A Systemic Methodological Framework for IS Research", *Proceedings of the Tenth Americas Conference on Information Systems AMCIS. New York (USA),* pp. 4.374-4.383, 2004.

[18] Baskerville, R., "Investigating Information Systems with Action Research", *Communications of the Association for Information Systems,* vol. 2, nº 19, pp. 1-32, 1999.

[19] Kitchenham, B., "Evaluating Software Engineering Methods and Tools. Part 1: The Evaluation Context and Evaluation Methods", *ACM Software Engineering Notes,* vol. 21, nº 1, pp. 11- 14, 1996.

[20] Ortega, M., Pérez, M. y Rojas, T., "Construction of a Systemic Quality Model for Evaluating a Software Product", *Software Quality Journal*, vol. 11, n° 3, pp. 219-242, 2003.

[21] Pérez, M., Domínguez, K., Mendoza, L. y Grimán, A., "Human Perspective in System Development Quality", *Proceedings of the Twelfth American Conference on Information Systems. AMCIS2006. Acapulco (México),* pp. 3.823-3.834, 2006.

[22] Kirchner, L. y Jung, J., "A Framework for the Evaluation of Meta-Modelling Tools", *The Electronic Journal Information Systems Evaluation*, vol. 10, n° 1, pp. 65-72, 2007.

X Jornadas de Innovación y Calidad del Software
Conferencia Iberoamericana de Calidad del Software

EOI Escuela de Negocios, Madrid, 24-25 de septiembre de 2008

Organizadas por el grupo de Calidad del Software de ATI (www.ati.es/gtcalidadsoft) en colaboración con EOI

Patrocinadores Premium:

Medio colaborador:

Colaboran:

Con el apoyo (Orden ITC/390/2007) institucional de: