Revista Española de Innovación, Calidad e Ingeniería del Software

REICIS

Volumen 7, No. 2, octubre, 2011

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

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

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

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

Contenidos

REICIS

Editorial

REICIS

Aunque en software la tecnología cambia continuamente (nuevos lenguajes, herramientas, arquitecturas, entornos, etc.), la necesidad de gestionar y controlar la calidad no cambia. En efecto, el interés de la industria, de los profesionales y de los investigadores puede irse centrando en conceptos nuevos y significativos como SOA (*Service Oriented Architecture*), *Cloud computing*, aspectos, aplicaciones para movilidad, sistemas empotrados, etc. Sin embargo, obtendremos pocas ventajas de estos cambios si, a la vez que la investigación tecnológica, no se abordan los métodos para lograr que la aplicación de estos nuevos conceptos vayan acompañados de sus nuevos métodos de gestión, de aseguramiento de calidad, etc. Es cierto que, en algunos casos, se ha comprobado que las técnicas ya existentes para desarrollo convencional de software sólo necesitan pequeños ajustes para seguir siendo útiles para estos nuevos proyectos. Pero, en otros, hay que crear nuevas formas de gestionar los distintos aspectos de la calidad e, incluso, redefinir o modelar el concepto de calidad aplicable.

En este sentido, lo fundamental es que se analicen e investiguen las consecuencias para la calidad de todos estos paradigmas novedosos para actuar en consecuencia. Con demasiada frecuencia, en el pasado, hemos visto cómo las nuevas tecnologías han irrumpido con fuerza en el mercado y en la investigación sin que se estudie cómo manejar los proyectos de desarrollo. Por ello, animamos a compañías, investigadores y profesionales a trabajar en estos campos de unión entre distintas disciplinas que, en algunas ocasiones, son inter y multidisciplinares para facilitar que los nuevos avances lleguen a todos en las condiciones exigibles de calidad y de facilidad de gestión.

Luis Fernández Sanz
Director

Presentación

REICIS

Este número de REICIS publica, tras el proceso de revisión de nuestro comité editorial, dos contribuciones remitidas por los autores como parte del proceso regular de recepción de artículos de la revista.

El primero de los trabajos publicados corresponde al titulado “Evidencia empírica sobre mejoras en productividad y calidad en enfoques MDD: un mapeo sistemático” procedente de un equipo de autores de la Universidad de Ciego de Ávila (Cuba) y de la Universidad de Alicante (España). En este artículo, Yulkeidi Martinez, Cristina Cachero y Santiago Meliá presentan los resultados de un estudio de la bibliografía científica centrada en las mejoras de calidad y productividad que supuestamente permite obtener seguir un enfoque de desarrollo de software dirigido por modelos (MDD: *Model Driven Development*). Su aportación permite, gracias al rigor de la técnica de revisión sistemática de las referencias bibliográficas, un análisis sólido de las evidencias documentadas por los investigadores en el mencionado enfoque de desarrollo de software.

El segundo trabajo se titula “Guía de pruebas de software para MoProSoft” y ha sido elaborado por Silvia Guardati y Alain Ponce del Instituto Tecnológico Autónomo de México. Tomando como guía de actuación el Modelo de Procesos para la Industria de Software (MoProSoft), los autores hacen una propuesta para la organización de las pruebas del software.

Finalmente, en la columna de Actualidad Invitada, Marcos Blanco, del CESJE (Centro de Excelencia de Software José de Espronceda, en Extremadura, España), nos acerca al enfoque y las experiencias de aseguramiento de calidad de software en proyectos para las Administraciones Públicas.

Luis Fernández Sanz

Evidencia empírica sobre mejoras en productividad y calidad en enfoques MDD: un mapeo sistemático

Yulkeidi Martínez
Universidad de Ciego de Ávila, Cuba
yulkeidi@gmail.com
Cristina Cachero, Santiago Meliá
Universidad de Alicante, España.
{ccachero, santi}@dlsi.ua.es

Resumen

Para el avance del desarrollo de software dirigido por modelos, es esencial proporcionar evidencias empíricas que corroboren o refuten las promesas de mejora asociadas a este paradigma desde su concepción. El objetivo de este trabajo es clasificar la evidencia empírica existente respecto de la mejora en productividad y calidad de las aplicaciones. Para ello hemos aplicado el proceso de mapeo sistemático, un tipo de estudio secundario diseñado específicamente para abordar este tipo de objetivos. Como resultado de este trabajo, hemos identificado asunciones que carecen a día de hoy de evidencia empírica. Por tanto, constituyen líneas de trabajo que se deben abordar para una mayor rigurosidad y consistencia de la disciplina. También hemos identificado áreas donde un análisis más exhaustivo podría ser de utilidad. El mapa resultante facilita la entrada de nuevos investigadores a este campo.

Palabras clave: Mapeo sistemático, ingeniería dirigida por modelos, calidad, productividad

Empiric evidence on productivity and quality improvements with MDD approaches: a systematic mapping

Abstract

In order to consolidate the progress in the development of the Model-Driven Development paradigm, it is essential to provide empirical evidence that either corroborates or refutes the promises of improvement that attached to this paradigm since its inception. The purpose of this paper is to classify the existing empirical evidence referred to improvements in productivity and quality of the applications. In order to achieve this goal, we have applied the systematic mapping process, a type of secondary study specifically devoted to carry out this kind of studies. As a result of this work, we identified assumptions based on today's lack of empirical evidence, and therefore lines of work are to be addressed to bring more rigor and consistency in discipline. We have also identified areas where further analysis could be useful. The resulting map facilitates the entry of new researchers to the field.

Key words: Systematic Mapping, Model Driven Development, Quality, Productivity

Martínez, Y., Cachero, C. y Meliá, S., "idencia empírica sobre mejoras en productividad y calidad en enfoques MDD: un mapeo sistemático", REICIS, vol. 7, no.2, 2011, pp. 6-28. Recibido: 21-2-2011; revisado: 11-7-2011; aceptado: 15-9-2011.

1. Introducción

El Desarrollo Dirigido por Modelos (MDD, del inglés *Model Driven Development*) es una aproximación al desarrollo de software basado en (a) la creación de modelos del sistema software a distintos niveles de abstracción y (b) su uso como base de un proceso de generación automática de código [1]. Entre las reivindicaciones de este paradigma de desarrollo se encuentran [2] [3] [4]:

1. Mayor simplicidad del proceso. El desarrollador se puede aislar de la complejidad tecnológica y centrarse en la estructura y comportamiento deseado de la aplicación.
2. Mejora de la productividad del proceso de desarrollo. El uso de modelos independientes de cómputo (CIM, *Computation-Independent Model*), modelos independientes de plataforma (PIM, *Platform-Independent Model*) y modelos de plataforma específica (PSM, *Platform-Specific Model*) permiten especificar el sistema a distintos niveles de abstracción y favorecen de este modo el reuso. La definición de transformaciones modelo a modelo y modelo a código automatiza gran parte del proceso de codificación.
3. Mejora de la calidad externa de la aplicación resultante (Funcionalidad, Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad) [5].

Sin embargo, y a pesar de las numerosas llamadas de atención por parte de la comunidad de Ingeniería del Software acerca de la necesidad de acompañar este tipo de afirmaciones con evidencias empíricas [6] [7], la gran mayoría de aportaciones en el campo del MDD sigue tomando la forma de nuevas metodologías, técnicas y herramientas que, aunque viables, no llegan a demostrar de una manera fiable su utilidad y ventajas respecto a sus predecesoras.

Con el fin de ayudar a que la comunidad proporcione estas evidencias empíricas, el área de experimentación en Ingeniería del Software ha desarrollado guías exhaustivas que ayudan a los investigadores en el proceso de obtención de datos fiables acerca de las ventajas o desventajas de los distintos métodos, técnicas o herramientas empleadas en la construcción de sistemas software [8] [9]. En ausencia de estos datos, se sigue corriendo el peligro de sostener conclusiones erróneas [10], perjudicando de esta manera tanto la toma

de decisiones en el ámbito empresarial [11] como la propia imagen de la disciplina, tal y como ya ha sucedido en el pasado [12].

Aún más importante, es necesario proporcionar un acceso rápido, claro y conciso a las evidencias empíricas de las que se dispone, de manera que ese conocimiento llegue a los encargados de decidir acerca de su adopción en la práctica. En el caso de MDD, esta falta de organización de la evidencia empírica, así como, cuando existe, su falta de relación con metodologías bien definidas, puede estar perjudicando su adopción por parte de las empresas; es bien sabido que la decisión de adoptar una nueva aproximación o utilizar nuevas herramientas en un proceso de desarrollo de software debería venir avalada por un proceso fiable y repetible, de manera que se maximicen las probabilidades de éxito de la implantación en la industria [13].

Con el objetivo de (a) atraer la atención de la comunidad de Ingeniería del Software sobre la falta de un acerbo empírico en el campo de MDD y (b) proporcionar un mapa conceptual que organice los datos que se han publicado hasta el momento, este trabajo presenta un mapeo sistemático [14] [15] de la evidencia empírica existente acerca de cómo MDD contribuye a mejorar la calidad de la aplicación resultante y la productividad del proceso de desarrollo. Esta evidencia se relaciona directamente con las reivindicaciones 2 y 3 presentadas al inicio de esta sección.

El trabajo está organizado como sigue: en la Sección 2 se presenta en detalles el proceso de del mapeo sistemático. En la sección 3 se presenta el análisis comparativo y se discuten los resultados del mapeo sistématico y sus limitaciones. Por último, en la sección 4 se presentan las conclusiones y trabajos futuros.

2. Mapeo Sistemático

La técnica de mapeo sistemático (*systematic mapping*) define un proceso y una estructura de informe que permite categorizar los resultados que han sido publicados hasta el momento en un área determinada [15]. El objetivo de un mapeo sistemático está en la clasificación, y está por tanto dirigido al análisis temático y a la identificación de los principales foros de publicación. Permite responder preguntas genéricas como ¿Qué es lo que se ha hecho hasta el momento en el campo X? Como limitación, este tipo de estudios no toma en consideración la calidad de los estudios incluidos. Una alternativa al mapeo

sistemático es la revisión sistemática [14], cuya fase de revisión de trabajos, mucho más rigurosa, permite establecer el estado de evidencia a través de la exhaustiva extracción de datos cuantitativos y estudios de meta-análisis, y por tanto responder a preguntas de investigación mucho más específicas. Estos dos tipos de estudios son complementarios y tienen como objetivo identificar los huecos de la investigación, por lo que un mapeo sistemático es considerado por muchos como un paso previo imprescindible para decidir en qué áreas concretas del campo es interesante abordar una revisión sistemática más detallada [16]. El proceso de mapeo sistemático seguido en la presente investigación se presenta en la figura 1.

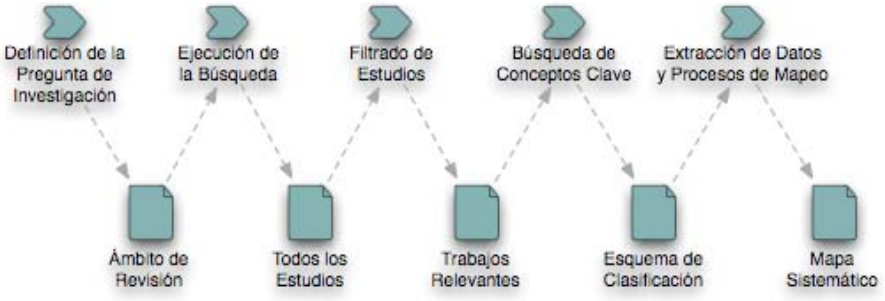

Figura 1. Proceso de mapeo sistemático.

2.1. Definición de la pregunta de investigación.

De acuerdo a [14], una pregunta de mapeo sistemático bien focalizada incluye cuatro partes:

Población:	Profesionales interesados en migrar hacia procesos de desarrollo basados en el paradigma MDD.
Factor de estudio:	Impacto de MDD sobre la calidad externa del software y la productividad del proceso.
Intervención en la comparación:	Marcos de trabajo, metodologías, herramientas, etc. basadas en el paradigma MDD que han servido de base para comparar productividad del proceso y/o calidad externa de las aplicaciones resultantes con respecto a otros paradigmas.
Resultado:	Grado de evidencia empírica existente en el campo.

Estos elementos nos han permitido definir las siguientes preguntas de investigación (PI):

- PI1: ¿Qué tipos de publicación ofrecen más datos empíricos sobre el impacto de MDD sobre la calidad y la productividad, y cómo ha cambiado la tendencia a lo largo del tiempo?
- PI2: ¿Qué características de calidad de producto (funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad [5]) y proceso han sido más investigadas desde un punto de vista empírico?
- PI3: ¿Qué tipo de estudio empírico (encuestas, casos de estudio, experimentos, meta-análisis) es el más usual a la hora de aseverar el impacto de MDD sobre características de calidad y productividad?
- PI4: ¿Qué tipo de enfoque de investigación (validación en entornos controlados o evaluación en entornos reales) es el más utilizado en el campo?

En base a estas preguntas, el objetivo de la investigación presentada en este artículo se puede resumir como: "identificar los estudios empíricos que se han realizado durante el período 2000-2010 sobre la mejora de la calidad del producto y la productividad del proceso mediante el uso de aproximaciones MDD".

2.2. Ejecución de la Búsqueda.

La cadena de búsqueda utilizada como base para la obtención de los trabajos relevantes ha sido: *empirical* AND *software* AND (*quality or performance*) AND (*"model driven"* OR *model-driven* OR MDD OR MDE OR MDA) AND (*experiment* OR *survey* OR *"case study"* OR meta-analysis). La búsqueda se ha centrado en los años 2001-2010. La elección del período temporal está motivada por el hecho de que el 2001 fue el año en que la OMG (*Object Management Group*) propuso la adopción de MDA (*Model Driven Architecture*) como estándar para las actividades involucradas en el MDD, aunque la guía oficial no fue publicada hasta junio 2003 [17].

Por otro lado, las fuentes de datos utilizadas han sido cuatro: *Google Scholar*, ACM, IEEE y *Springer*. Esta selección de fuentes es hasta cierto punto redundante, ya que *Google Scholar*, motor de búsqueda líder en el seno de la comunidad científica de investigadores académicos, indexa un gran número de fuentes de documentación técnica, entre las que se encuentran ACM, IEEE y *Springer*, que a su vez son los tres foros más significativos en la Ingeniería del Software. *Google Scholar* recupera además documentos que no aparecen en bibliotecas digitales organizadas, y que sin embargo forman parte del acerbo científico de

las distintas disciplinas, por lo que consideramos que es un complemento importante en la elaboración de mapeos y revisiones sistemáticas [18]. No menos importante, *Google Scholar* muestra los documentos ordenados en función de la importancia del foro de publicación en el que se encuentra el artículo, la relevancia de sus autores o la frecuencia con la que es citado el escrito. Por tanto su inclusión nos ayuda a garantizar que estamos incluyendo los trabajos más relevantes para la comunidad científica.

Los resultados de ejecutar nuestra cadena de búsqueda en *Google Scholar* fueron 896 artículos, que incluyen, como ya hemos comentado, tanto publicaciones indizadas en las librerías digitales más importantes en el área de Ingeniería del Software (incluidas ACM, IEEE y *Springer)* como literatura gris publicada on-line en las Webs de las universidades (informes técnicos, tesis de grado, maestrías, tesis doctorales, etc.). Una revisión preliminar de los resultados nos permitió constatar que las publicaciones relevantes para nuestro estudio se encontraban concentradas en los primeros puestos de la búsqueda, por lo que se decidió limitar el análisis más exhaustivo de las mismas a los 300 primeros trabajos.

Nuestro segundo paso de búsqueda consistió en adaptar la cadena de búsqueda a la idiosincrasia de los buscadores específicos de ACM, IEEE y *Springer*. La ejecución de la búsqueda en estos motores específicos arrojó 99 resultados (ACM), 3136 resultados (IEEE) y 317 resultados (*Springer)* respectivamente. Nuevamente, dado que los tres los buscadores ordenan por relevancia de la publicación, se analizaron los 100 resultados más relevantes de cada buscador, lo que nos da un total de 599 publicaciones analizadas. En la Tabla 1 se presenta un resumen de estos resultados.

Buscador	***Google Scholar***	**ACM**	**IEEE**	***Springer***	**Total**
Resultados de la Consulta	896	99	3136	317	4448
Trabajos analizados	300	99	100	100	599
Trabajos candidatos	187	76	52	39	354
Trabajos Relevantes	40	12	6	6	64
Coincidencias con *Google Scholar*	-	9	1	4	14
Total Trabajos Relevantes	40	3	5	2	50

Tabla 1. Resultados de la búsqueda antes y después de eliminar duplicados.

2.3. Filtrado de Estudios

El protocolo inicial de revisión definido para la selección de los estudios primarios se ha formulado basado en los siguientes criterios de inclusión/exclusión:

Inclusión: libros, documentos, informes técnicos y la literatura gris que describe los estudios empíricos sobre calidad o productividad en MDD, incluso aunque solo se tenga acceso al resumen del mismo.

Exclusión:

1. Artículos que no reportan estudios empíricos acerca de las mejoras en calidad o productividad cuando se utiliza una aproximación MDD. Esto implica dejar de lado cualquier trabajo centrado en justificar la mera viabilidad de la propuesta (sin comprobación empírica de las mejoras que introducen en cuanto a productividad y calidad externa del producto final).
2. Discusiones teóricas, revisiones y clasificaciones, así como propuestas de modelos de calidad que no vienen acompañados de un estudio empírico.
3. Estudios que se centran en evaluar/mejorar la calidad de los modelos y/o transformaciones que intervienen en las aproximaciones MDD, salvo que lo hagan en relación a su impacto sobre la calidad del producto final.
4. Estudios empíricos sobre aproximaciones MDD que estudian factores de contexto o datos subjetivos (e.g. los que evalúan los factores que influyen en su adopción exitosa, opinión subjetiva de desarrolladores, etc.).

2.3.1 Fiabilidad del criterio de inclusión

Para evaluar la fiabilidad de los criterios de inclusión/exclusión de los estudios relevantes, y por tanto incrementar las posibilidades de obtener resultados fiables e independientes del evaluador, se seleccionó una submuestra de la población, consistente en las 100 primeras referencias arrojadas por la búsqueda inicial en *Google Scholar*, lo que supone un 11,16% de la población total de los resultados arrojados por este buscador, y un 33% de los resultados finalmente analizados del mismo. Tras el establecimiento de los criterios de revisión, el título y el resumen de dichas referencias fueron utilizados para clasificar los trabajos de manera independiente por dos revisores: E1 y E2. La fiabilidad inter-evaluador se calculó mediante el estadístico Kappa de *Cohen* [19]. El grado de fiabilidad arrojado por el estadístico fue satisfactorio (Kappa=0,811, ver tabla 2). Este grado de acuerdo indica la existencia de una base de criterios suficientemente clara y que no denota divergencias significativas entre los revisores [20].

		E1		Total
		0	**1**	
E2	**0**	86	2	**88**
	1	2	10	**12**

Tabla 2. Tabla de Contingencia. 2da fase del proceso de inclusión/exclusión (título + resumen)

Durante la fase de conciliación de diferencias entre los evaluadores resultó especialmente interesante constatar la poca homogeneidad en cuanto al formato de informes de resultados empíricos, así como el uso tan libre que se sigue haciendo de palabras como "caso de estudio" o "experimento" en la literatura de Ingeniería del Software, a pesar de las numerosas llamadas de alerta al respecto [20]. En la práctica, esas palabras se refieren en muchas ocasiones a meros estudios de viabilidad de la propuesta, lo que complica notablemente la obtención de resultados relevantes en una búsqueda como la planteada en el presente artículo.

Los criterios validados fueron aplicados a los resultados devueltos por los cuatro motores de búsqueda utilizados como fuentes de datos para el mapeo sistemático. Finalmente se incluyeron en el estudio un total de 50 trabajos: 40 de ellos aparecían en los resultados de *Google Scholar*, 3 en ACM, 5 en IEEE y 2 en *Springer*. En la tabla 1 se presentan los datos pormenorizados. Analizando dicha tabla se aprecia que el grado de coincidencia de resultados entre el buscador genérico y los buscadores específicos apenas alcanza un 55%, lo que demuestra la complementariedad de *Google Scholar* y el resto de motores de búsqueda [18]. Además, es interesante notar cómo IEEE aparece como la fuente de información peor cubierta por *Google Scholar* (ver tabla 1).

2.4. Definición del esquema de clasificación.

Una vez seleccionados los trabajos relevantes se definieron, en base a los objetivos del estudio, tres tipos de clasificaciones (ver figura 2):

- Medidas evaluadas: de calidad de proceso (productividad) y de calidad de producto (fucionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad). Esta última clasificación se basa en el modelo de calidad presentado en la norma ISO/IEEE 9126 [5].
- Tipo de estudio empírico realizado [21] [22]: caso de estudio, experimento, encuesta o meta-análisis.

- Enfoques de investigación: validación en entornos controlados o evaluación en entornos reales [23].

Además, para cada estudio relevante se recopiló información referente al año de publicación y tipo de publicación mediante el que fue diseminado.

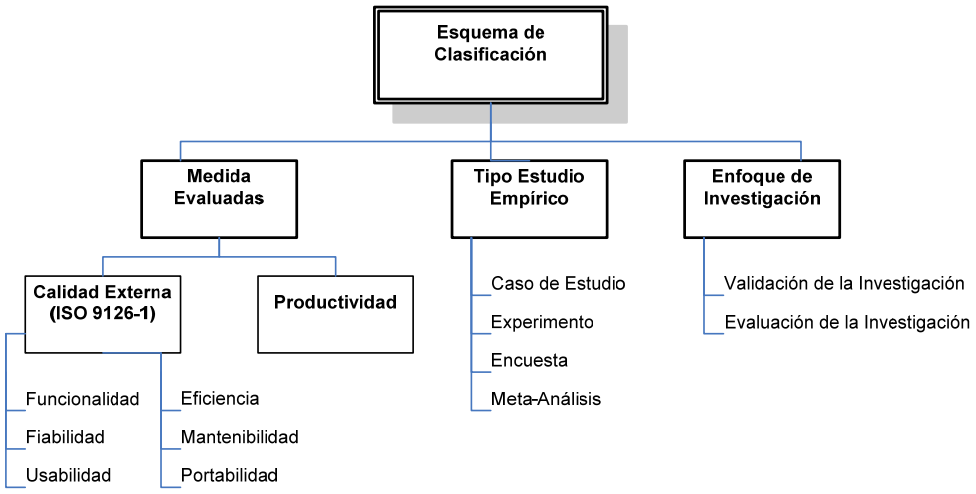

Figura 2. Esquema de clasificación.

2.5. Extracción de datos y Mapeo sistemático.

Tras definir el sistema de clasificación, el último paso del mapeo sistemático consiste en la extracción de datos y el proceso de mapeo de las distintas dimensiones. El resultado completo de esta actividad se muestra en el Apéndice A. El resultado sintetizado de nuestro estudio se puede observar de manera gráfica en el diagrama de burbuja de la figura 3, que visualiza: (1) la relación entre el tipo de estudio empírico y el estudio de medidas de calidad impactadas por el uso del paradigma MDD y (2) la frecuencia de estudios empíricos por años de publicación.

La figura 3 ilustra básicamente dos diagramas de dispersión XY con burbujas en las intersecciones de categoría, que permite tener en cuenta varias categorías al mismo tiempo y da una visión general rápida de un campo de estudio, proporcionando un mapa visual [15]. En esta visualización de los resultados, el tamaño de una burbuja es proporcional al número de artículos que están en el par de categorías que correspondan a la burbuja de las coordenadas. Cuando un trabajo ha afectado más de una categoría (e.g. un meta-análisis de más de una característica de calidad, o presentación de más de un tipo de estudio empírico),

el trabajo ha sido contabilizado en todas las características. De este modo, los porcentajes dan una visión más real sobre la cantidad de esfuerzo invertido en cada una de las dimensiones de calidad incluidas en el estudio.

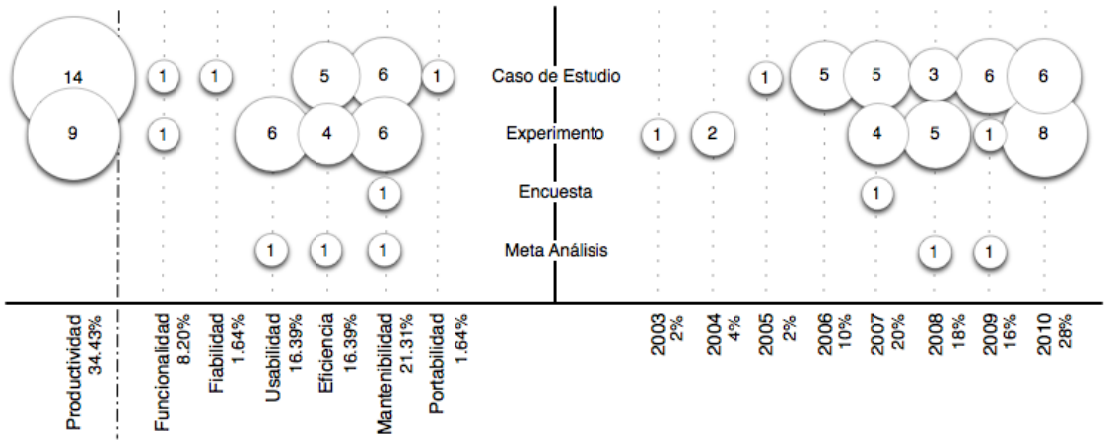

Figura 3. Diagrama de burbuja. Visualización del mapeo sistemático.

De igual forma, en la figura 4 se puede observar la distribución de trabajos por tipo de publicación y por enfoque de investigación. Del total de publicaciones incluidas en el mapeo (50), 21 pertenecen a revistas, 20 fueron presentadas en conferencias y 9 provienen de otros tipos de publicaciones como libros, tesis o informes técnicos. Por otro lado, 28 consistieron en validaciones en ambiente controlado, frente a 22 evaluaciones (en entornos de uso real) del paradigma.

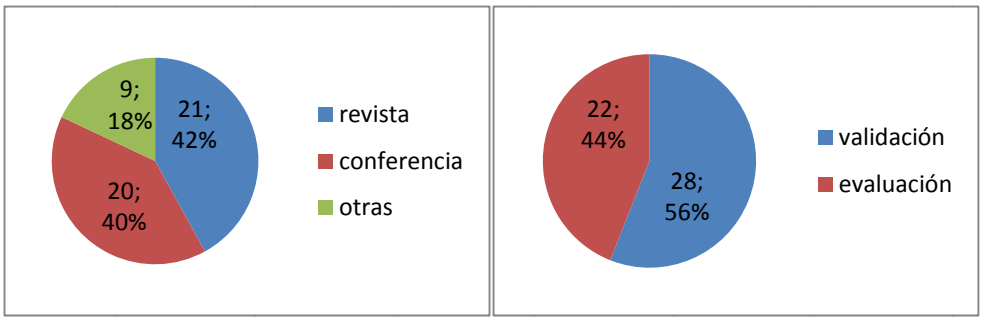

Figura 4. Diagramas Circulares. Artículos por tipo de publicación y por enfoque de investigación.

3. Análisis comparativo y discusión

A continuación damos respuesta a las preguntas de investigación formuladas en la Sección 2.1. a través de los resultados obtenidos.

- PI1: Tras un análisis pormenorizado de los resultados obtenidos, se puede apreciar que la publicación de estudios empíricos en el ámbito de MDD presenta una

tendencia al alza, en consonancia con lo que está ocurriendo en otros campos de la Ingeniería del Software. Un indicador de esta tendencia es que aproximadamente 28% de los estudios empíricos realizados en MDD se concentran en el año 2010. Los foros preferidos para este tipo de trabajos son revistas (42%), lo que indica la relevancia otorgada por los investigadores a los resultados obtenidos. Otro dato interesante arrojado por el estudio es que, de las 599 publicaciones candidatas a ser incluidas en el estudio, solo 50 (un 8,3%) cumplieron los criterios para ser finalmente incluidas en el análisis, lo que, dado la especificidad de la búsqueda, por un lado demuestra el uso tan libre que se hace de los términos empíricos en el área, y por otro es un indicador de la falta de madurez empírica de MDD, lo que nos lleva a pensar que puede ser pronto todavía para embarcarse en una revisión sistemática de dicha evidencia empírica. En caso de realizarse, los campos más prometedores serían el impacto de MDD en productividad y mantenibilidad.

- PI2: La mantenibilidad (ver figura 3) con un 21,31% de los resultados centrados en ella, es la característica de calidad externa que más ha sido investigada en el enfoque MDD. No lejos se encuentran la eficiencia y la usabilidad, con un 16,39% cada una. El resto de las medidas han sido estudiadas en menor escala, lo que plantea una oportunidad para futuras tesis y trabajos de investigación. Por otro lado, el estudio empírico de la productividad, como medida de calidad del proceso de desarrollo (eficiencia de proceso) ha tenido un auge representativo en los últimos 3 años, con un 34,43% de los resultados empíricos centrados en ella. Este dato parece sugerir que el trabajo empírico en MDD se orienta a demostrar la calidad de proceso más que la calidad de producto, lo que es consistente con la preponderancia de reivindicaciones de mejora del proceso entre las razones esgrimidas por los investigadores para la adopción del paradigma.
- PI3: Como se puede observar en la figura 3, los casos de estudio y los experimentos resultan ser los estudios empíricos con mayor predominio en las investigaciones actuales. Este es un dato prometedor, ya que estos estudios, donde se mide lo que hacen los sujetos, en lugar de medir lo que dicen que hacen, ofrecen un mayor grado de fiabilidad. Además, es interesante constatar cómo la usabilidad ha sido estudiada

exclusivamente mediante experimentos, cuyos resultados son los más fiables a costa de un alcance necesariamente más limitado.

- PI4: En cuanto a los enfoques de investigación más utilizados, los números (ver figura 4) indican que la mayoría de las investigaciones actuales sobre el uso del enfoque MDD han demostrado la aplicabilidad de su propuesta a través de validaciones (estudios en entornos sintéticos). En concreto, 28 publicaciones de las 50 (56%) han utilizado este enfoque, lo que aumenta la fiabilidad de los resultados pero no permite identificar con exactitud los beneficios e inconvenientes de su aplicación en entornos reales de desarrollo (al contrario de lo que ocurre con la evaluación de la investigación) [23]. Además, durante el último año incluido en el estudio (2010) se ha incrementado la cantidad de estudios evaluados empíricamente tanto en entornos sintéticos como en la industria (ver Apéndice A, tabla A.1), lo que nos permite afirmar que la comunidad investigadora está siendo cada vez más cuidadosa en propiciar investigaciones claras, reproducibles y con garantías de aplicación industrial.

3.1 Limitaciones del estudio

La principal limitación de este mapeo sistemático es haber analizado sólo 599 publicaciones del total de 4448 resultados obtenidos a las consultas realizadas en los diferentes buscadores (ver tabla 1). Aunque esta decisión sin duda perjudica la cobertura del estudio, creemos que esta desventaja se ve paliada por el uso de los buscadores más relevantes del campo *(Google Scholar*, ACM, IEEE y *Springer*) y por la ordenación que hacen dichos buscadores de los resultados en función de su relevancia para la búsqueda.

La segunda limitación, debida a la propia filosofía del mapeo sistemático, es la calidad de los estudios incorporados. Esta limitación se podría haber evitado realizando una revisión sistemática de dichos trabajos. Sin embargo, el bajo número de publicaciones verdaderamente pertinentes (50), nos hace pensar que es todavía pronto para este tipo de evaluaciones de calidad.

Otra posible limitación de este tipo de estudios es la posibilidad de errar la clasificación por el uso ambiguo que hacen los autores de conceptos como, en nuestro caso, experimento o caso de estudio. Relacionado con esto, hemos ratificado lo que ya advertían otros autores [15] acerca de los resúmenes de los artículos, que a menudo son engañosos y

carecen de información importante. En este estudio se ha limitado el impacto de estos dos riesgos mediante una aproximación conservadora al proceso de inclusión/exclusión de estudios, que ha implicado la lectura de cuantas partes del artículo hayan sido necesarias hasta poder resolver la duda de si incluir o excluir el estudio.

4. Conclusiones y trabajos futuros

Es un hecho que los estudios empíricos en el área del MDD se han incrementado sustancialmente en los últimos años, lo cual permite ya contar con ciertas evidencias del impacto de este paradigma sobre la calidad y productividad de los productos de software, más allá de resultados anecdóticos.

Sin embargo, el porcentaje de estudios que realmente proporcionan evidencia empírica acerca de mejoras de calidad y productividad en MDD sigue siendo muy bajo (solo la productividad del paradigma ofrece un número de estudios comparativamente significativo), lo que contrasta con otras disciplinas e incluso con otras áreas de Ingeniería del Software [24]. Por tanto, sigue siendo relevante enfatizar la necesidad de que la comunidad de MDD dedique una cantidad sustancial de esfuerzo a la comprobación empírica de sus aseveraciones que permita la posterior realización de revisiones sistemáticas y meta-análisis rigurosos.

De manera general, los resultados más relevantes reportados por los estudios empíricos relacionados con la productividad de MDD se pueden resumir como sigue:

- Los estudios que validan el uso de MDD en entornos académicos reportan una productividad de 2 a 9 veces superior a la obtenida con otros paradigmas de desarrollo [25]. La productividad puede llegar a ser hasta 20 veces superior según se va aumentando el tamaño del proyecto de desarrollo. Estos resultados contrastan con los reportados por experimentos en entornos industriales [26], donde los resultados son mucho más heterogéneos, y van desde los que directamente reportan una pérdida de productividad de un 10% [27] a los que coinciden con los estudios académicos y reportan ganancias que oscilan entre un 20 % y un 35 % de productividad [27] [28] [29] [30]. Como principal limitación a estos resultados, los informes en entornos industriales se basan en estudios a pequeña escala. Las principales razones esgrimidas para justificar las pérdidas de productividad se

relacionaban directamente con el uso de herramientas inmaduras y altos costos de modelado, que puede llegar a ser tan complejo como programar directamente con un lenguaje tradicional de tercera generación.

- Por otro lado, se ha constatado cómo el uso de herramientas, librerías especializadas, etc. de soporte a MDD reduce la curva de aprendizaje y el esfuerzo de entrenamiento con este paradigma, lo que a su vez incide en la productividad del paradigma [31].
- Existen evidencias de que el enfoque MDD presenta deficiencias a la hora de expresar las reglas de diseño arquitectónicas de la aplicación, lo que actúa en detrimento de la productividad y la calidad de la aplicación final [32].

Con respecto a la mantenibilidad, los artículos incluidos en este estudio reportan que el tiempo necesario para evaluar el impacto de un cambio es substancialmente más corto (con un decremento que, en los estudios revisados, ronda el 37%) si se usa una visualización gráfica (base del MDD) en comparación con una textual (no MDD) [33]. Además, usando aproximaciones MDD completas para el proceso de mantenimiento se mejora aún más la mantenibilidad [34]. Estos datos contrastan en cierta medida con lo reportado en [35], donde se expone que el uso de UML como lenguaje de modelado para tareas de mantenimiento no tiene un impacto significativo en el tiempo necesario para realizar un cambio, pues también se debe contar el tiempo para poner al día la documentación de UML. Sin embargo, por lo que se refiere a la exactitud funcional de los cambios (introducción de errores en el software durante el cambio), UML ha demostrado un impacto positivo a la hora de mejorar la calidad del código, incluso si los desarrolladores no están muy familiarizados con su uso. Otros estudios [36] [37] [38] presentan experimentos para medir el efecto del reuso, la complejidad y el acoplamiento de objetos en la mantenibilidad del software, aunque sin comparar distintas aproximaciones:

- La reusabilidad del software aumenta en detrimento de la simplicidad y mantenibilidad de algunos artefactos [36].
- Existe correlación entre la complejidad de un diagrama y las facilidades de comprensión y modificación del mismo [37].

- De igual forma, el acoplamiento entre objetos está fuertemente correlacionado con las facilidades de comprensión y modificación de expresiones OCL que pueden actuar como aserciones en el código final [38].

Si nos centramos en dominios concretos, una comparación del Modelado Orientado a Aspectos (MOA) *vs* la Codificación Orientada a Aspectos, los estudios experimentales y cuantitativos realizados en [39] reportan que las aproximaciones MOA generan una aplicación más pequeña, menos compleja y más modular. Además, en algunos casos el desarrollo basado en MOA acorta el ciclo de mantenimiento.

En cuanto al impacto de MDD sobre la eficiencia, funcionalidad, fiabilidad, usabilidad y portabilidad, la literatura provee resultados en su mayor parte anecdóticos, poco contrastados, lo que hace muy difícil su generalización. Los estudios se centran principalmente en presentar experiencias donde prácticas específicas (una determinada herramienta, aproximación, técnica, modelo, etc.) han mostrado sus bondades a la hora de mejorar la calidad del producto final.

A pesar de que nuestro estudio abarca el período comprendido entre los años 2001 y 2010, el trabajo empírico en el campo sigue dando frutos, y en los últimos meses se han seguido publicando estudios empíricos relevantes para nuestra investigación. En concreto, en [40] [41] se ha estudiado el impacto de los factores sociales, técnicos y orgánicos en el éxito o fracaso de la aproximación MDD en la industria. Particularmente se han encontrado una correlación positiva entre un entorno con mejores conocimientos organizativos y mejor comunicación en el equipo de desarrollo y las variables de productividad y mantenibilidad del proceso MDD, con un incremento de 66,7% y 73,7% respectivamente. También se han recopilado datos que parecen sugerir que el uso de modelos en la comprensión de problemas con alto nivel de abstracción incide en el aumento tanto de la productividad (hasta un 72,2%) como en la mantenibilidad (hasta un 73,4%). Otras conclusiones de estos estudios inciden en la importancia del entrenamiento y la educación para alcanzar las promesas de este paradigma.

Con el mapeo sistemático presentado en este trabajo se han conseguido identificar los principales trabajos de investigación empírica publicados hasta el momento en los principales foros científicos. Este tipo de estudios es básico de cara a facilitar la apertura

del campo a nuevos investigadores. Además, este estudio ha puesto de manifiesto las enormes posibilidades de trabajo en este campo para la comunidad de Ingeniería del Software empírica con respecto al impacto del uso de aproximaciones basadas en el paradigma MDD sobre la calidad tanto de producto como de proceso.

De especial relevancia resulta constatar los resultados tan dispares encontrados en función de si los estudios han sido realizados en un entorno académico o industrial y, sobre todo, en función del tamaño del proyecto abordado. Otras variables que, hasta lo que alcanza nuestro conocimiento, no han sido suficientemente consideradas y que podrían influir en los resultados son los tipos de aplicaciones abordadas (e.g. si se tratan de aplicaciones intensivas en datos, transaccionales, etc.) o la experiencia de los desarrolladores. Por último, sorprende no encontrar ningún meta-análisis de productividad del paradigma, pese a que sí existen meta-análisis de otras variables con menos contribuciones. Pensamos que esto es debido, por un lado, a la disparidad en cuanto a la calidad de los informes publicados, que a veces presentan carencias que hacen imposible evaluar la calidad del estudio y agregar los datos, y por otro a la heterogeneidad de los contextos y premisas sobre las que se han construido los estudios, lo que igualmente complica su comparación y la agregación de sus resultados. El uso de plantillas y paquetes experimentales estandarizados ayudaría sin duda a paliar estos problemas y a agilizar la obtención de datos de calidad.

Agradecimientos

A Jesús Pardillo, por sus útiles comentarios durante la elaboración de este trabajo.

Referencias

[1] Muñoz, J. y Pelechano, V., “MDA vs Factorías de Software”. En: Estévez, A., Pelechano V. y Vallecillo, A. (eds.), *Actas del II Taller sobre Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (DSDM 2005). Granada (Spain), 13 de Septiembre*, pp. 1, 2005.

[2] Macías, M., “Modelamiento basado en el Dominio: Estado del Arte”, *Revista de las I Jornadas de Ingeniería de Software 2004*, vol. 1, nº 1, pp. 33-41, 2004.

[3] Selic, B., "Model-Driven Development: Its Essence and Opportunities". En: IEEE Computer Society Press (eds.), *Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06). Gyeongju (Korea), April 24-26,* pp. 313-319, 2006.

[4] López, E., González, M., López, M. y Iduñate, E.L., "Proceso de Desarrollo de Software Mediante Herramientas MDA", *Revista Iberoamericana de Sistemas, Cibernética e Informática*, vol. 3, nº 2, pp. 6-10, 2006.

[5] ISO/IEC 9126-1, *Software engineering - Product quality Part 1: Quality model*, ISO, 2001.

[6] Glass, R., "The Realities of Software Technology Payoffs", *Communications of the ACM* , vol. 42, nº 2, pp. 74-79, 1999.

[7] Pickard, M., *Combining Empirical Results in Software Engineering*. Technical Report vol. 1, Keele University, England, 2004.

[8] Kitchenham, B., Linkman, S. y Law, D., "DESMET: a methodology for evaluating software engineering methods and tools", *Computing and Control Engineering Journal*, vol. 8, nº 3, pp. 120-126, 1997.

[9] Guerini, M. L., *Revisión de resultados experimentales en técnicas de prueba y de educción de conocimientos*. Tesis de Magister en Ingeniería de Software, Universidad Politécnica de Madrid, España, 2007.

[10] Juristo N. y Moreno A., *Basics of Software Engineering Experimentation*, Kluwer Academic Publisher, 2001.

[11] Picó, R., *Análisis funcional de la gestión de almacén y producción en openbravo ERP y su aplicación docente*. Proyecto fin de Carrera, Escuela Técnica Superior de Informática Aplicada, Universidad Politécnica de Valencia, España, 2010.

[12] Vanderburg, G., Real Software Engineering, video/mp4 presentations of Lone Star Ruby Conference 2010, 2010. Disponible en: http://confreaks.net/videos/282-lsrc2010-real-software-engineering) [Consulta 17/10/2010].

[13] Kriter Software SL., Los diez factores que garantizan el éxito de la implantación ERP, 2007. Disponible en: http://www.kriter.net/item/es/software-empresa-erp-kriter-identifica-los-diez-factores-que-garantizan-el-exito-de-la-implantacion/18/25/ [Consulta 30/09/2010].

[14] Kitchenham, B. y Charters, S., *Guidelines for performing systematic literature reviews in software engineering*. Technical Report EBSE-2007-01, School of Computer Science and Mathematics, Keele University, Inglaterra, 2007.

[15] Petersen, K., Feldt, R., Mujtaba, S. y Mattsson, M., "Systematic mapping studies in software engineering". En: Visaggio, G., Baldassarre, M.T., Linkman, S. y Turner, M. BCS eWIC (eds.), *Proceedings of 12th International Conference on Evaluation and Assessment in Software Engineering (EASE). University of Bari (Italy), June 26 - 27*, pp. 71-80, 2008.

[16] Mujtaba, S., Petersen, K., Feldt, R. y Mattsson, M., "Software product line variability: A systematic mapping study", School of Engineering, Blekinge Inst. of Technology, 2008. Disponible en: sites.google.com/site/drfeldt/mujtaba_apsec08_sysmap_spl_variabili.pdf [Consulta 14/10/2010].

[17] Object Managament Group, *Overview and guide to OMG's architecture* (*MDA Guide v1.0.1)* OMG, 2003. Disponible en: www.omg.org/cgi-bin/doc?omg/03-06-01 [Consulta 15/12/2009].

[18] Noruzi, A., "Google Scholar: The new generation of citation indexes", *Libri/Copenhagen Journal*, vol. 55, nº 4, pp. 170-180, 2005.

[19] Gwet, K., "Inter-Rater Reliability: Dependency on Trait Prevalence and Marginal Homogeneity", *Series of Statistical Methods for Inter-Rater Reliability Assessment*, vol. 2, nº 2, pp. 1-9, 2002.

[20] Fleiss, J.L., *Statistical methods for rates and proportions*, John Wiley, 1981.

[21] Kish, L., "Some statisticals problems in research design", *American Sociological Review*, vol. 24, nº 3, pp. 328-338, 1959.

[22] Mendes, E. y Mosley, N., *Web Engineering*, Springer-Verlag New York Inc, 2006.

[23] Wieringa, R., Maiden, N., Mead, N. y Rolland, C., "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion", *Journal Requirements Engineering*, vol. 11, nº 1, pp. 102-107, 2006.

[24] Zelkowitz, M., "An update to experimental models for validating computer technology", *The Journal of Systems and Software*, vol. 82, nº 3, pp. 373–376, 2009.

[25] Diaz, O. Villoria F.M., "Generating blogs out of product catalogues: An MDE approach", *The Journal of Systems and Software*, vol. 83, nº 10, pp. 1970-1982, 2010.

[26] Mohagheghi, P. y Dehlen, V., "Where is the proof? - A review of experiences from applying MDE in industry". En: Schieferdecker, I. y Hartman, A. (eds.), *Proceedings of Model Driven Architecture: Foundations and Applications: 4th European Conference, (ECMDA-FA 2008)*, Berlin (Germany), June 09-12, pp. 432–443, 2008.

[27] MODELWARE D5.3-1 Industrial ROI, Assessment and Feedback-Master Document. Revision 2.2, 2006. Disponible en: www.modelware-ist.org [Consulta 28/09/2010].

[28] The Middleware Company, Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) Approach: Productivity Analysis (White Paper), Report by the Middleware Company on behalf of Compuware, 2003. Disponible en: www.omg.org/mda/mda_files/MDA_Comparison-TMC_final.pdf [Consulta 28/09/2010].

[29] MacDonald, A., Russell, D. y Atchison, B., "Model-Driven Development within a Legacy System: an Industry Experience Report". En: IEEE Computer Society Press (eds.), *Proceedings of Australian Software Engineering Conference (ASWEC 2005). Brisbane (Australia), 29 March-1 April, 2005*, pp. 14–22, 2005.

[30] Lange, C. y Chaudron, M., "Interactive Views to Improve the Comprehension of UML Models - An Experimental Validation". En: *IEEE* Computer Society Press (eds.), *Proceedings of the 15th IEEE International Conference on Program Comprehension (ICPC '07). Banff, Alberta (Canada), June 26-29*, pp. 221-230, 2007.

[31] Zhu, L., Gorton I., Liu Y. y Bao Bui N., "Model driven benchmark generation for web services". En: Di Nitto, E., Hall, R.J., Han, J., Han, Y., Polini, A., Sandkuhl, K. y Zisman A. (eds.), *Proceedings of the 2006 International Workshop on Service-Oriented Software Engineering (SOSE 2006). Shanghai (China), May 27-28*, pp. 33-39, 2006.

[32] Mattsson, A., Lundell, B., Lings, B y Fitzgerald, B., "Linking Model-Driven Development and Software Architecture: A Case Study", *IEEE Transactions on Software Engineering*, vol. 35 nº 1, pp. 83-93, 2009.

[33] Mellegård, N. y Staron, M., "Improving Efficiency of Change Impact Assessment Using Graphical Requirement Specifications: An Experiment". En: Babar, M.A., Vierimaa, M. y Oivo, M. (eds.), *Proceedings of Product-Focused Software Process Improvement 11th International Conference (PROFES 2010). Limerick (Ireland), June 21-23*, pp. 336–350, 2010.

[34] Goldschmidt, T., Reussner, R. y Winzen, J., "A Case Study Evaluation of Maintainability and Performance of Persistency Techniques". En: Schäfer, W., Dwyer, M.B. y Gruhn, V. (eds.), *Proceedings of the 30th International Conference on Software Engineering (ICSE 2008). Leipzig (Germany), May 10-18*, pp. 401-410, 2008.

[35] Dzidek W.J., Arisholm, E. y Briand, L.C.A, "Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance", *IEEE Transactions on Software Engineering*, vol. 34, nº 3, pp. 407-432, 2008.

[36] Lucrédio, D., "Evaluating the impact of MDD in software reuse", 2009, Disponible en: www.les.inf.puc-rio.br/ [Consulta 03/11/2010].

[37] Manso, M. E., Cruz-Lemus, J. A. Genero, M. y Piattini, M., "Empirical Validation of Measures for UML Class Diagrams: A Meta-Analysis Study". En: Chaudron, M.R.V. (ed.), *Proceedings of the Workshop in Model Software Engineering: Workshops and Symposia in MODELS 2008). Toulouse (France), September 28-October 3*, pp. 303-313, 2008.

[38] Reynoso, L., Genero, M. and Manso, E., "Measuring Object Coupling in OCL Expressions: A Cognitive Theory-Based Approach". En: Daponte, P. (ed.), *Proceedings of the 23rd IEEE Instrumentation and Measurement Technology Conference (IMTC 2006). Sorrento (Italy), April 24-27*, pp. 1087–1092, 2007.

[39] Hovsepyan, A., Scandariato, R. Van Baelen, S., Berbers, Y. y Joosen, W., "From Aspect-Oriented Models to Aspect-Oriented Code? The Maintenance Perspective". En: Jézéquel, J. M. y Südholt, M. (eds.), *Proceedings of the 9th International Conference on Aspect-Oriented Software Development (AOSD 2010). Rennes and Saint Malo (France), March 15 -19*, pp. 85-96, 2010.

[40] Hutchinson, J., Rouncefield, M. y Whittle, J., "Model-driven engineering practices in industry". En: Taylor, R.N., Gall, H. y Medvidovic, N. (eds.), *Proceeding of the 33rd International Conference on Software Engineering (ICSE 2011). Waikiki, Honolulu (Hawaii), May 21-28*, pp. 633-642, 2011.

[41] Hutchinson, J., Whittle, J., Rouncefield, M. y Kristoffersen, S., "Empirical assessment of MDE in industry". En: Taylor, R.N., Gall, H. y Medvidovic, N. (eds.), *Proceeding of the 33rd International Conference on Software Engineering (ICSE 2011). Waikiki, Honolulu (Hawaii), May 21-28*, pp. 471-480, 2011.

Apéndice A: Publicaciones relevantes clasificadas

Publicación	Año	Fuente	Tipo de estudio	Medida de Calidad	Enfoque de Investig.
The value of conceptual modeling in database development: An experimental investigation	2003	GS	E	Productividad	V
The Impact of Software Reuse and Incremental Development on the Quality of Large Systems	2004	GS	E	Productividad, Mantenibilidad	Ev
Configuring real-time aspects in component middleware	2004	GS	E	Eficiencia	Ev
Alfresco Content Display	2005	GS	CE	Productividad	Ev
From code centric to model centric software engineering: practical case study of MDD infusion in a systems integration company	2006	GS	CE	Productividad	Ev
Applying model-driven development to distributed real-time and embedded avionics systems	2006	GS	CE	Productividad	V
Analysis of crosscutting in model transformations	2006	GS	CE	Mantenibilidad	V
Applying system execution modeling tools to evaluate enterprise distributed real-time and embedded system QoS	2006	GS	CE	Eficiencia	V
Model driven benchmark generation for web services	2006	ACM	CE	Productividad	Ev
A model-driven architecture approach using explicit stakeholder quality requirement models for building dependable information systems	2007	GS	CE	Fiabilidad	V
Quicker: A model-driven QoS mapping tool for QoS-enabled component middleware	2007	GS	CE	Funcionalidad	V
An activity-based quality model for maintainability	2007	GS	CE	Mantenibilidad	V
A Case Study on Model-Driven and Conventional Software Development: The Palladio Editor	2007	GS	CE	Productividad	Ev
mTurnpike: a Model-driven Framework for Domain Specific Software Development	2007	GS	E	Eficiencia	V
Building measure-based prediction models for UML class diagram maintainability	2007	GS	E	Mantenibilidad	V
Measuring Object Coupling in OCL Expressions: A Cognitive Theory-Based Approach	2007	GS	E	Mantenibilidad	V
Reliable Effects Screening: A Distributed Continuous Quality Assurance Process for Monitoring Performance Degradation in Evolving Software Systems	2007	GS	CE	Eficiencia	Ev
Interactive views to improve the comprehension of UML models-an experimental validation	2007	IEEE	E	Productividad	V
Survey of traceability approaches in model-driven engineering	2007	IEEE	S	Mantenibilidad	Ev
Where is the proof?-a review of experiences from applying MDE in industry	2008	GS	CE	Productividad	V
Transitioning from code-centric to model-driven industrial projects–empirical studies in industry and academia	2008	GS	E	Productividad	V
Factors affecting developers' use of MDSD in the Healthcare Domain: Evaluation from the MPOWER Project	2008	GS	E	Funcionalidad, Usabilidad, Productividad	Ev
Usability evaluation of user interfaces generated with a model-driven architecture tool	2008	GS	E	Usabilidad	V
Analyzing the Influence of Certain Factors on the Acceptance of a Model-based Measurement Procedure in Practice: An Empirical Study	2008	GS	E	Usabilidad	V
A realistic empirical evaluation of the costs and benefits of uml in software maintenance	2008	IEEE	E	Mantenibilidad	Ev

A case study evaluation of maintainability and performance of persistency techniques	2008	IEEE	CE	Productividad Mantenibilidad	Ev
A Model-Based Framework for Security Policy Specification, Deployment and Testing	2008	Springer	CE	Eficiencia	Ev
Model-Driven Performance Evaluation for Service Engineering	2008	Springer	MA	Eficiencia	Ev
Empirical investigations of model size, complexity and effort in a large scale, distributed model driven development process	2009	GS	CE	Productividad	V
Embedded System Construction–Evaluation of Model-Driven and Component-Based Development Approaches	2009	GS	CE	Mantenibilidad	Ev
Evaluating the Correctness and Effectiveness of a Middleware QoS Configuration Process in Distributed Real-time and Embedded Systems	2009	GS	CE	Portabilidad	Ev
Level of detail in UML models and its impact on model comprehension: A controlled experiment	2009	GS	E	Usabilidad	V
Empirical Validation of Measures for UML Class Diagrams: A Meta-Analysis Study	2009	GS	MA	Usabilidad, Mantenibilidad	V
MDA-based tool chain for web services development	2009	GS	CE	Productividad	V
A Comparative Case Study of Model Driven Development vs Traditional Development: The Tortoise or the Hare	2009	GS	CE	Productividad	V
Linking model-driven development and software architecture: A case study	2009	IEEE	CE	Productividad	V
Model-driven development for early aspects, Volume 52, Issue 3	2010	GS	CE	Mantenibilidad, Eficiencia	Ev
Distribution of Effort Among Software Development Artefacts: An Initial Case Study	2010	GS	CE	Productividad	V
The impact of structural complexity on the understandability of UML statechart diagrams	2010	GS	E	Usabilidad. Productividad	V
MOOGLE: A model search engine	2010	GS	E	Productividad	Ev
An Empirical Investigation of the Utility of 'pre-CIM'models	2010	GS	E	Productividad	Ev
Towards to the validation of a usability evaluation method for model-driven web development	2010	GS	E	Eficiencia, Usabilidad	Ev
Productivity Analysis of the Distributed QoS Modeling Language	2010	GS	CE	Productividad	V
Usability evaluation of multi-device/platform user interfaces generated by model-driven engineering	2010	GS	E	Usabilidad	V
Improving Efficiency of Change Impact Assessment Using Graphical Requirement Specifications: An Experiment	2010	GS	E	Eficiencia, Mantenibilidad	V
Moppet: A Model-Driven Performance Engineering Framework for Wireless Sensor Networks	2010	GS	CE	Mantenibilidad	Ev
2010 36th EUROMICRO Conference on Sofware Engineering and Advanced Applications (The Impact of Model Driven Development on the Software Architecture Process)	2010	GS	CE	Eficiencia	V
Enterprise systems development: Impact of various software development methodologies	2010	GS	CE	Productividad	V
From aspect-oriented models to aspect-oriented code?: the maintenance perspective	2010	ACM	E	Mantenibilidad	Ev
Generating blogs out of product catalogues: An MDE approach	2010	ACM	E	Productividad	Ev

Leyenda:

GS: *Google Scholar*	**CE**: Caso de Estudio **E**: Experimento **MA**: Meta análisis	**S**: Encuesta (*Survey*) **Ev**: Evaluación **V**: Validación

Tabla A.1. Mapa sistemático

Guía de pruebas de software para MoProSoft

Silvia Guardati, Alain Ponce
Departamento Académico de Computación, Maestría en Tecnologías de Información y Administración
Instituto Tecnológico Autónomo de México
guardati@itam.mx, alain@tecnoimplanta.com

Resumen

Existen varios modelos de procesos de software reconocidos y utilizados por la industria del software. En general, los mismos ofrecen el marco para que las empresas dedicadas al desarrollo de software puedan definir procesos propios para regir sus actividades. Es decir, especifican qué hacer. Sin embargo, estos modelos no dicen cómo hacerlo. En cuanto a las pruebas, si bien está demostrado que son fundamentales para asegurar la calidad del software, en las PYMES no siempre se les da la atención necesaria. En este trabajo se presenta una Guía de Pruebas de Software (GPS) para MoProSoft que complementa al modelo, brindando a las PYMES la información y las herramientas útiles para llevar a cabo las pruebas.

Palabras clave: modelos de procesos de software, MoProSoft, pruebas, PYME.

Software testing guide for MoProSoft

Abstract

There are several software process models recognized and used by the software industry. All these models offer a set of valuable processes which allow companies to standardize their activities and guarantee software products of greater quality. Nevertheless, it is worth noting that the process models indicate the *know-what*, rather than the *know-how*. In relation to software testing, companies -specially Small and Medium Enterprises (SMEs)-, do not spend the time and effort required to guarantee the quality of the products throughout their entire lifecycle. In this paper, we present a Software Testing Guide to MoProSoft that complements this model, providing information and useful elements for carrying out testing duties in all categories identified by MoProSoft.

Key words: Software Process Models, MoProSoft, Testing, SME.

Guardati, S. y Ponce, A. "Guía de pruebas de software para MoProSoft", REICIS, vol. 7, no.2, 2011, pp. 28-49. Recibido: 15-6-2010; revisado:25-8-2010; aceptado: 18-9-2011.

1. Antecedentes

Existen varios modelos de procesos de software reconocidos y utilizados por la industria del software. Estos modelos ofrecen un marco de procesos que permite a las empresas definir procesos propios para regir sus actividades y, de esta manera, lograr mayor eficiencia. Entre los modelos más conocidos se puede mencionar *Capability Maturity Model Integrated* (CMMI) [1] y los estándares ISO/IEC 12207 [2] e ISO/IEC 15504 [3]. En Latinoamérica, sobresalen Modelo de Procesos para la Industria de Software (MoProSoft) [4] y *Melhoria de Processo do Software Brasileiro* (MPS.Br)[1]. Los modelos MoProSoft y MPS.Br están orientados a las PYMES. En el caso de Brasil, las PYMES representan el 70% [5] del total de sus empresas, mientras que en México[2] casi el 90%. Es importante destacar que MoProSoft, además de convertirse en norma mexicana (NMX-I-059/04-NYCE-2005)[3], fue tomado como uno de los documentos de trabajo para la elaboración de la norma internacional *Lifecycle Profiles For Very Small Entities* - ISO/IEC 29110, a cargo del ISO/IEC JTC1/SC7 *Working Group* 24 [6].

Todos estos modelos ofrecen un conjunto de procesos valiosos para que las empresas puedan estandarizar sus actividades y asegurar, por medio del seguimiento de los mismos, productos de software de mayor calidad. Sin embargo, los modelos de procesos especifican el *know-what*, pero no el *know-how*. En organizaciones grandes, esta característica no afecta ya que las mismas cuentan con áreas responsables de la definición de procesos y, en general, hacen uso de metodologías, estándares, tecnologías, etc. Es decir, los modelos les ofrecen la referencia para ordenar e institucionalizar muchos de los elementos que aplican en sus actividades cotidianas. En el caso de las PYMES la situación es distinta [7]. Aproximadamente el 75% de las empresas de software, a nivel mundial, tienen menos de 25 empleados [8] y, por lo tanto, cada empleado desempeña varios roles dentro de un proyecto y el ritmo de las actividades está dirigido por los compromisos y no tanto por los procesos a seguir. Por esta misma razón, es costoso para una organización de este tipo disponer de personal que se dedique a la definición de los procesos y de todos los elementos requeridos por los mismos. En el caso particular de México, la sexta estrategia del Programa de Desarrollo de la Industria del Software (PROSOFT) está dedicada a

[1] http://es.scribd.com/doc/65438748/MPS-BR-Guia-Geral-V1-2
[2] http://www.economia.gob.mx
[3] http://www.nyce.org.mx/

mejorar la capacidad de procesos de las empresas del sector de servicios de las tecnologías de información (TI). MoProSoft es un resultado de esta iniciativa.

Respecto a las pruebas, las empresas (muchas veces por cuestiones de tiempo y presupuesto) no las llevan a cabo con el rigor esperado. En ciertos casos, aplican las pruebas sólo en la fase de implementación y sin enfocarse en las estrategias de la propia empresa. Las consecuencias son costosas para la organización: tiempo extra de programación, pérdida de confianza del cliente, incremento del tiempo total de desarrollo, entre otras. En un mercado tan competitivo como el de las PYMES, estas consecuencias pueden ocasionar el fracaso. En [7] se menciona que las empresas pequeñas requieren de más apoyo para la adopción de estándares, solicitando concretamente guías con ejemplos y plantillas.

Considerando lo expresado más arriba, la Guía de Pruebas de Software (GPS) complementa el modelo MoProSoft, ofreciendo una guía completa para el desarrollo de las actividades de pruebas en las tres categorías del modelo. GPS es el resultado de la aplicación de pruebas de software y de años de experiencia en la industria del software en México. La guía se elaboró con base en la experiencia obtenida en una empresa del sector servicios de tecnologías de información, de más de 2000 empleados, en la cual se realizan pruebas en cada etapa del desarrollo de productos de software. Posteriormente se aplicó con buenos resultados en dos PYMES con menos de 10 empleados, ambas del sector servicios de TI.

2. Modelo MoProSoft

Definido como “Modelo de Procesos para la Industria de Software (MoProSoft) en México que fomente la estandarización de su operación a través de la incorporación de las mejores prácticas en gestión e ingeniería de software. La adopción del modelo permitirá elevar la capacidad de las organizaciones para ofrecer servicios con calidad y alcanzar niveles internacionales de competitividad” [4].

Para cumplir con su objetivo, MoProSoft está organizado en tres categorías. Éstas cubren todas las áreas críticas de una organización dedicada al desarrollo de software: Alta Dirección, Gerencia y Operación. En la figura 1 se presenta un esquema con cada una de las categorías de procesos, así como de los procesos que las integran. Para cada uno de

estos el modelo señala nombre, categoría, propósito, actividades, objetivos, indicadores de objetivos, roles, entradas, salidas, productos internos, prácticas, referencias bibliográficas, subprocesos (si hubiera) y procesos relacionados [9].

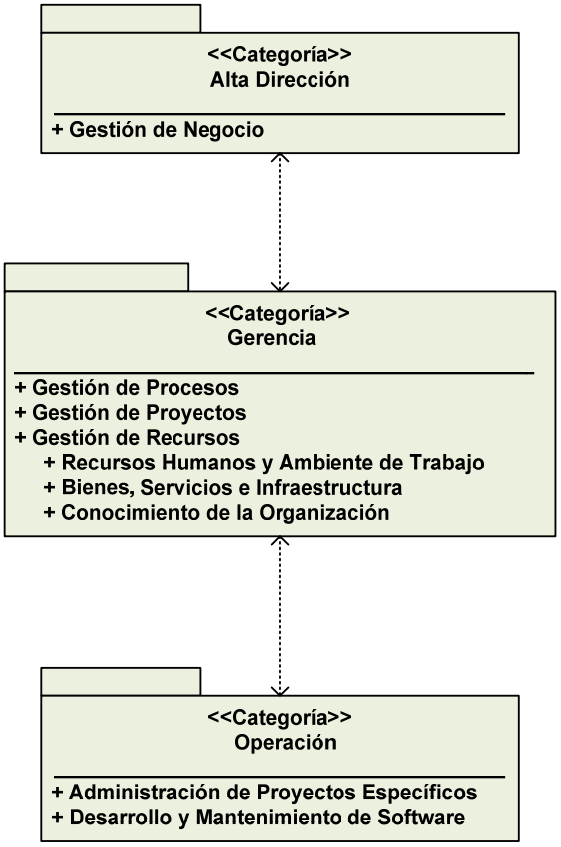

Figura 1. Categorías de Procesos de MoProSoft [4]

- Categoría Alta Dirección. Se enfoca a las prácticas relacionadas con la gestión del negocio. También proporciona los lineamientos a los procesos de la categoría Gerencia y recibe la retroalimentación requerida a partir de la información generada por estos.
- Categoría Gerencia. Se enfoca a las prácticas relacionadas con la gestión de procesos, proyectos y recursos de acuerdo a los lineamientos establecidos por la categoría Alta Dirección. Además, proporciona los elementos necesarios a los procesos de la categoría Operación, recibe y evalúa la información generada por estos y comunica los resultados a la categoría Alta Dirección.

- Categoría Operación. Se enfoca a las prácticas de los proyectos de desarrollo y mantenimiento de software. Lleva a cabo las actividades según los elementos dados por Gerencia y comunica a ésta sus resultados.

MoProSoft está dirigido a las empresas pequeñas y medianas dedicadas al desarrollo y/o mantenimiento de software. Provee un conjunto de procesos, los cuales pueden ser adoptados, ajustándolos a las características propias de cada organización. El modelo es fácil de entender y de aplicar y su implementación no es costosa. Sin embargo, requiere que quien lo implemente defina el cómo, lo cual implica un esfuerzo considerable, sobre todo en las organizaciones que no disponen de empleados dedicados a tal fin. GPS es una guía que complementa al modelo, proporcionando elementos útiles (dicen cómo) en el área de pruebas.

3. Guía de Pruebas de Software para MoProSoft

La Guía de Pruebas de Software (GPS) está estructurada de acuerdo a las categorías de MoProSoft y cubre las actividades cuyos resultados pueden mejorarse por medio de las pruebas. Es decir, se eligieron aquellas en las cuales las pruebas pueden intervenir para asegurar la calidad del producto generado, siguiendo el modelo.

Si bien las pruebas se usan para demostrar que el producto de software hace lo que se supone debe hacer y no hace nada que no esté previsto [10], es importante su consideración en todos los niveles de la organización. Es decir, para que las mismas puedan aplicarse con éxito durante el ciclo de desarrollo, deben ser contempladas inicialmente como estratégicas por parte de directivos y responsables de la gestión de los proyectos. Es por ello, que las pruebas deben formar parte de cualquier actividad desarrollada en una organización ya que son el medio para asegurar la calidad.

En algunos casos, las actividades de pruebas de la guía se identifican con el mismo nombre que las actividades de MoProSoft que están complementando. En otros, GPS propone sus propias actividades. A continuación se describe la aportación de GPS en cada una de las categorías del modelo. Por cuestiones de espacio, sólo en algunas actividades se incluyen ejemplos. Una versión completa de GPS puede encontrarse en [11].

3.1. Categoría Alta Dirección

Las pruebas son clave para garantizar la calidad de cualquier proyecto de software, por lo que la Alta Dirección debe conocer y estar comprometida con la aplicación de las mismas desde la perspectiva del negocio. El proceso de esta categoría, *Gestión de Negocio*, contempla tres actividades de las cuales la guía cubre las dos que considera apropiadas para incluir pruebas. En la tabla 1, primera columna, se listan las actividades (Ai) de MoProSoft que cuentan con una actividad de prueba definida por GPS, segunda columna.

Actividades MoProSoft Proceso Gestión de Negocio	**Actividades GPS**
A1. Planificación estratégica	1. Entender la situación actual
	2. Desarrollar o actualizar objetivos y estrategias
	3. Verificar el Plan Estratégico
A3. Valoración y mejora continua	Análisis de la información y evaluación del desempeño

Tabla 1. Proceso Gestión de Negocio - Actividades de MoProSoft cubiertas por GPS

3.1.1. Actividad MoProSoft: Planificación Estratégica (A1)

GPS complementa esta actividad con otras tres (ver tabla 1) que permiten incluir las pruebas durante la ejecución de la misma. A continuación se describen brevemente cada una de ellas.

Con la actividad “entender la situación actual” la guía ayuda a determinar la calidad de los productos que se están entregando, las posibles áreas de mejora, así como aquellas fortalezas que se deben mantener.

La actividad “desarrollar o actualizar objetivos y estrategias” se lleva a cabo una vez concluido el análisis de la situación actual. Se deben establecer objetivos para el área de pruebas así como las estrategias para lograrlos. GPS describe cómo realizar esta actividad y presenta algunos ejemplos útiles.

Por último, en “verificar el Plan Estratégico”, GPS propone pruebas estáticas sobre la documentación del Plan Estratégico para asegurar un material completo y correcto.

3.1.2. Actividad MoProSoft: Valoración y Mejora Continua (A3)

Para realizar un “análisis de la información y evaluación del desempeño”, GPS sugiere la recolección de indicadores en todos los proyectos y da una lista de los que resultan de mayor interés. El registro de indicadores se retoma en la categoría Operación.

3.2. Categoría Gerencia

En esta categoría la guía define las actividades de pruebas que deben considerarse dentro de los procesos *Gestión de Procesos* y *Gestión de Recursos*. En las tablas 2 y 3 se presenta la relación entre actividades de los procesos mencionados y los elementos aportados por GPS a cada uno de ellos, respectivamente.

Actividades MoProSoft Proceso Gestión de Procesos	**Actividades GPS**
A1. Planificación	Establecer o actualizar la definición de elementos de procesos

Tabla 2. Proceso Gestión de Procesos - Actividades de MoProSoft cubiertas por GPS

3.2.1. Actividad MoProSoft: Planificación (A1)

GPS propone “establecer o actualizar la definición de elementos de procesos”, que consiste en definir una librería de procesos enfocados a las pruebas, así como su nomenclatura.

Actividades MoProSoft Proceso Gestión de Recursos	**Actividades GPS**
A1. Planificación de recursos	Generar o actualizar el Plan de Adquisiciones y Capacitación
A3. Investigación de tendencias tecnológicas	Generar propuestas tecnológicas

Tabla 3. Proceso Gestión de Recursos - Actividades de MoProSoft cubiertas por GPS

3.2.2. Actividad MoProSoft: Planificación de Recursos (A1)

GPS indica “generar o actualizar el Plan de Adquisiciones y Capacitación”. Para las adquisiciones plantea algunas recomendaciones, así como una lista de insumos a tener en cuenta para la realización de las pruebas. Con respecto a la capacitación, GPS señala la necesidad de definir un plan de capacitación en el área de pruebas para que los involucrados adquieran el conocimiento requerido de acuerdo a los objetivos y estrategia de la empresa. GPS es en sí misma un medio para lograr esta capacitación. La misma puede complementarse con cursos sobre herramientas comerciales para la administración y ejecución automática de pruebas.

3.2.3. Actividad MoProSoft: Investigación de Tendencias Tecnológicas (A3)

GPS, por medio de la actividad “generar propuestas tecnológicas”, menciona y explica las tecnologías ligadas a las pruebas, así como la importancia de buscar un equilibrio entre costo/beneficio en cada una de ellas.

3.3. Categoría Operación

Las pruebas buscan asegurar, durante cada una de las fases de desarrollo, que el producto generado sea de alta calidad. También promueve el registro y control de indicadores de los proyectos. Los datos obtenidos serán de gran utilidad para la gerencia y para directivos para tomar decisiones correctamente informadas. En la categoría se encuentran los procesos Administración de Proyectos Específicos y Desarrollo y Mantenimiento de Software, por lo que ofrece un amplio espectro de aplicación de las pruebas.

El proceso de Administración de Proyectos Específicos incluye cuatro actividades y GPS aporta en dos de ellas. En la tabla 4 se muestra la relación entre actividades MoProSoft y GPS: en 3.3.1 y 3.3.2 se describen las correspondientes aportaciones de GPS.

Actividades MoProSoft Proceso Administración de Proyectos Específicos	**Actividades GPS**
A1. Planificación	1. Identificar ciclos y actividades
	2. Determinar el tiempo estimado para cada actividad
	3. Formar el equipo de trabajo
	4. Generar el calendario de trabajo
A4. Cierre	Formalizar la terminación del ciclo y del proyecto

Tabla 4. Proceso Administración de Proyectos Específicos - Actividades de MoProSoft cubiertas por GPS

Actividades MoProSoft Proceso Desarrollo y Mantenimiento de Software	**Actividades GPS**
A2. Realización de la fase de requisitos	1. Documentar o modificar la especificación de requisitos
	2. Verificar la especificación de requisitos
	3. Elaborar o modificar Plan de Pruebas de Sistema
	4. Planear pruebas de aceptación (NI)
A3. Realización de la fase de análisis y diseño	1. Generar la descripción de la estructura interna del sistema y del registro de rastreo
	2. Elaborar o modificar Plan de Pruebas de Integración
A4. Realización de la fase de construcción	Definir y aplicar pruebas unitarias a los componentes
Creación de los casos de prueba (NI)	1. Definir y crear casos y escenarios de pruebas (NI)
	2. Crear datos de pruebas (NI)
A5. Realización de la fase de integración y pruebas	1. Realizar integración y pruebas
	2. Realizar las pruebas de sistema
	3. Realizar las pruebas de aceptación (NI)
A6. Realización de la fase de cierre	1. Decisión de finalización del proyecto (NI)
	2. Manual de mantenimiento

Tabla 5. Proceso Desarrollo y Mantenimiento de Software - Actividades de MoProSoft cubiertas por GPS y actividades adicionales

El proceso Desarrollo y Mantenimiento de Software se compone por uno o varios ciclos de desarrollo. A su vez, cada ciclo se divide en fases: Inicio, Requerimientos, Análisis y Diseño, Construcción, Integración y Pruebas, y Cierre. GPS aporta en cinco de ellas. En la tabla 5 se presenta la relación entre actividades MoProSoft y GPS. Algunas actividades se acompañan con NI para indicar que la misma es propuesta por GPS y no está incluida en MoProSoft. En las secciones 3.3.3 a 3.3.8 se describen las aportaciones de GPS.

3.3.1 Actividad MoProSoft: Planificación (A1)

GPS complementa esta actividad con otras cuatro (ver tabla 4) que permiten incluir las pruebas durante la ejecución de la misma. A continuación se describen brevemente cada una de ellas.

Tipo de prueba	Menos de 3 ciclos	De 3 a 10 ciclos	Más de 10 ciclos
Revisión de documentos			X
Pruebas de integración de componentes	X		
Pruebas de humo	X		
Pruebas de sistema	X		
Pruebas de regresión	X		
Pruebas de aceptación de usuario		X	

Tabla 6. Cantidad de ciclos de pruebas según el tipo de prueba

El número de ciclos se define de acuerdo a los objetivos de la organización y a las actividades de pruebas que se harán en el proyecto. Por medio de la actividad “identificar ciclos y actividades” GPS propone, a partir de la experiencia, una escala de ciclos de prueba según los tipos de pruebas que se aplicarán (ver tabla 6). En cuanto a las actividades, la guía presenta y explica las principales orientadas a asegurar la calidad del producto de software. Asimismo, distingue entre actividades que deben aplicarse en todos los proyectos (se mencionan creación y revisión de documentos de pruebas, pruebas de integración de componentes, criterios de aceptación y detención, entre otras) y aquellas que sólo se aplican en casos especiales (se mencionan ejecutar pruebas con los usuarios finales en proyectos de gobierno, realizar pruebas de conectividad con dispositivos o sistemas externos, realizar pruebas de estrés, entre otras).

El tiempo para cada actividad depende de la experiencia de la empresa, del tamaño del proyecto y de los requisitos específicos de éste. A través de la actividad "determinar el tiempo estimado para cada actividad" GPS plantea la necesidad de que cada organización registre los tiempos de pruebas de todos sus proyectos, de tal manera que forme una base consistente de información que le permita hacer buenas estimaciones en proyectos futuros.

El equipo de trabajo está formado por una o más personas, pudiendo incluir personal externo. En general, las PYMES no cuentan con personal dedicado exclusivamente a las pruebas, sino que los roles se distribuyen entre los miembros del equipo. En la actividad "formar el equipo de trabajo" GPS completa la lista de roles señalados por MoProSoft, indicando para cada uno de ellos las capacidades requeridas así como las funciones que desempeña dentro del equipo de pruebas, destacando la importancia del trabajo cooperativo [12].

Una vez identificadas las actividades, estimados los tiempos y formado el grupo de trabajo, se procede a "generar el calendario de trabajo" que incluye fecha de inicio y de terminación de cada actividad, así como la secuencia y la dependencia entre ellas. GPS ofrece varios elementos que apoyan esta actividad. Uno de ellos es una escala de tiempo requerido para la corrección de defectos (actividad que no depende de los encargados de las pruebas) que puede tomarse como base hasta tanto se disponga de datos propios (ver tabla 7). También, destaca aspectos del negocio a tener en cuenta, como la rotación de personal y las ventas en tiempos cortos. Por último, guía sobre los aspectos relacionados a las pruebas que deben considerarse en el calendario global del proyecto.

Dificultad de la actividad	**Importancia del defecto encontrado**			
	Baja	**Media**	**Alta**	**Crítica**
Fácil	12 horas	4 horas	2 horas	1 hora
Media	24 horas	10 horas	4 horas	2 horas
Difícil	36 horas	16 horas	8 horas	6 horas

Tabla 7. Acuerdo de servicio para resolución de defectos

3.3.2 Actividad MoProSoft: Cierre (A4)

GPS indica "formalizar la terminación del ciclo y del proyecto" estableciendo criterios para aprobar el cierre de ambos. Se recomiendan criterios como porcentaje de requisitos implementados exitosamente o porcentajes de requisitos alcanzados por ciclo.

3.3.3 Actividad MoProSoft: Realización de la fase de Requisitos (A2)

GPS complementa esta actividad con otras cuatro (ver tabla 5) que permiten incluir las pruebas durante su ejecución. A continuación se describen brevemente cada una de ellas.

La especificación de requisitos es un documento formal con todas las funcionalidades que debe contar el sistema. Considerando que existen distintas maneras de clasificar los requisitos, en la actividad "documentar o modificar la especificación de requisitos" GPS propone como llevar a cabo esta tarea. Además, recomienda que tanto los desarrolladores como el equipo de pruebas sigan la misma tipificación.

La actividad "verificar la especificación de requisitos" se lleva a cabo sobre el resultado del paso anterior, y consta de dos partes. La primera, identificar todos los requisitos de manera única, sirve para tener una visión completa de los mismos, así como de su severidad y de los datos relevantes a cada uno de ellos. Además, facilita la administración de las actividades de pruebas ejecutadas sobre los mismos, contribuye a asegurar que todos los requisitos tienen asociada alguna prueba (cobertura total) y permite formar una base de requisitos única y administrada para su uso posterior. GPS propone un estándar para llevar a cabo la identificación única de los requisitos. En la figura 2 se presentan los pasos a seguir, según GPS. En la tabla 8 se presenta un ejemplo del estándar de identificación propuesto.

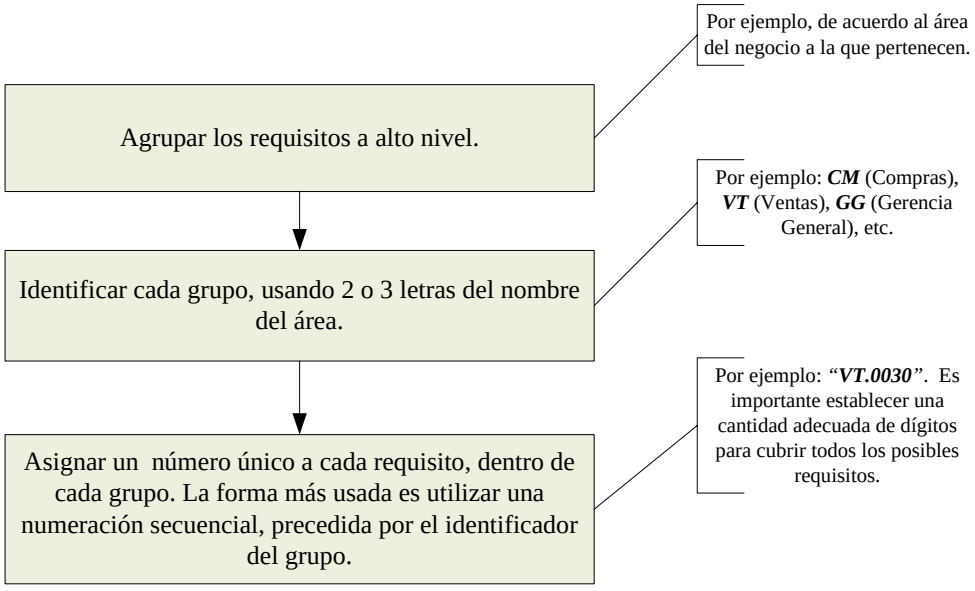

Figura 2. Pasos a seguir para establecer una identificación única de requisitos

La segunda parte de la verificación consiste en revisar los requisitos (pruebas estáticas) para asegurarse que cumplen con lo establecido por la organización y que los mismos están redactados claramente. Para lograr el primer objetivo, GPS presenta elementos a tener en cuenta desde tres perspectivas: la del contenido del documento de requisitos, la de las capacidades (tecnológicas, de recursos, etc.) y la del negocio. Las revisiones se llevan a cabo por miembros de distintos niveles de la organización y pueden realizarse utilizando diversas técnicas. GPS lista y explica las principales. Por último, GPS señala revisar la redacción de los requisitos por medio de técnicas grupales para asegurar que los mismos no contengan elementos incompletos o ambiguos y que todos los términos están claramente definidos.

Id. Negocio	Área de Negocio	Número	Id. requisito	Descripción requisito	Comentario
GN	GENERAL	0001	GN.0001	Usar la imagen de la empresa en todas las pantallas.	Las pantallas deben contener el logo y los colores de la empresa.
GN	GENERAL	0002	GN.0002	El sistema debe mostrar una pantalla de inicio indicando versión, fecha de creación y nombres …	Los nombres se pueden obtener del área Legal, del archivo "Nombre y Marcas Legales.doc".
GN	GENERAL	0003	GN.0003	Se deben manejar 3 niveles de usuarios...	Nivel 1: Administrador… Nivel 2: Ventas … Nivel 3: Compras …
GN	GENERAL	0004	GN.0004	El sistema debe operar con cualquiera de los sistemas operativos: …	El sistema no tiene que operar sobre …
VT	VENTAS	001	VT.001	Cada venta debe contener la siguiente información: clave empleado, nombre cliente, …	El número de factura electrónica debe ser el mismo que el impreso …

Tabla 8. Ejemplo de identificadores de requisitos

En general, las PYMES tienden a plantear las pruebas de sistema como un único periodo de tiempo con fecha de inicio y de fin. Sin embargo, es importante desglosar el plan con el fin de determinar el alcance y la cobertura de las pruebas y, en consecuencia, el grado de confiabilidad de las mismas. Por medio de la **actividad elaborar o modificar Plan de Pruebas de Sistema**, GPS presenta y explica los principales elementos a considerar en el Plan de Pruebas. En el caso (deseable) de que el plan haya sido elaborado

durante la actividad Planificación (A1), sólo se revisará y modificará, si corresponde, atendiendo las recomendaciones dadas.

Cuando el sistema lo justifique será necesario obtener la aceptación explícita del usuario por medio de **planear pruebas de aceptación** de usuario (no incluida en MoProSoft). En el Plan de Pruebas se debe tener en cuenta que en éstas participa el cliente. En la guía se mencionan y explican algunas tareas necesarias para llevarlas a cabo. Por ejemplo, modificar casos de prueba de sistema para aplicarlos en pruebas de aceptación, preparar el ambiente de ejecución para acercarlo al entorno de producción, administrar datos que son afectados en los diferentes escenarios de prueba.

3.3.4 Actividad MoProSoft: Realización de la fase de Análisis y Diseño (A3)

GPS complementa esta actividad con otras dos (ver tabla 5) que permiten incluir las pruebas durante la ejecución de la misma. A continuación se describen brevemente cada una de ellas.

Id. requisito	**Descripción requisito**	**Id. Requisito desglosado**	**Desglose del requisito (estructura interna)**
GN.0001	Usar la imagen de la empresa en todas las pantallas.	GN.0001.001	Existirá una clase “Imagen Empresa” que representará la imagen (logo y colores) de la empresa.
		GN.0001.002	Las pantallas podrán usar la clase “Imagen Empresa” para heredar sus características.
GN.0002	…	GN.0002.001	…
GN.0003	Se deben manejar 3 niveles de usuarios...	GN.0003.001	Crear un menú de nivel 1 para el Administrador.
		GN.0003.002	Crear un menú de nivel 2 para Ventas.
		GN.0003.003	Crear un menú de nivel 3 para Compras.
		GN.0003.004	Crear una pantalla de firma digital para seleccionar el menú a aplicar.

Tabla 9. Ejemplo de requisitos desglosados

Para la actividad “generar la descripción de la estructura interna del sistema y del registro de rastreo” GPS señala cómo identificar de manera única los nuevos requisitos obtenidos a partir del desglose de los iniciales y cómo realizar la revisión de esta actividad. En la tabla 9 se presentan los requisitos desglosados correspondientes a los de la tabla 8.

Con esta especificación se puede generar el registro de rastreo, al cual se le agregará, a medida que se avance en el proyecto, los tipos de pruebas a ejecutar, los casos de pruebas y la cobertura de estos. Un ejemplo de una matriz de rastreo se muestra en la tabla 10.

Matriz de rastreo **v.002** **Actualizada:**

Requisitos		**Casos de Prueba**					
Requisito	**Sub requisito**	**Caso 1**	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**	**Caso 6**
GN.0001	GN.0001.001						
	GN.0001.002						
GN.0002	GN.0002.001						
GN.0003	GN.0003.001						
	GN.0003.002						
	GN.0003.003						
	GN.0003.004						

Tabla 10. Ejemplo de matriz de rastreo

Con la actividad "elaborar o modificar Plan de Pruebas de Integración" GPS recomienda considerar las pruebas que permitan asegurar la interacción entre los distintos módulos del sistema y con sistemas externos, el adecuado uso de las interfaces externas del sistema y el manejo de errores en todas las transacciones probadas. Para llevar a cabo estas pruebas es necesario realizar un conjunto de actividades que las complementen, como por ejemplo la administración de las versiones de cada pieza de software y la creación de escenarios de prueba. GPS presenta y explica las principales. En el caso (deseable) de que el plan haya sido elaborado durante la actividad Planificación (A1), sólo se revisará y modificará, si corresponde, atendiendo las recomendaciones dadas.

3.3.5 Actividad MoProSoft: Realización de la fase de Construcción (A4)

GPS indica "definir y aplicar pruebas unitarias a los componentes". El objetivo de las pruebas unitarias es que el desarrollador compruebe el buen funcionamiento del componente. GPS propone la aplicación de dos técnicas para soportar esta actividad: el control de versiones y el registro de las pruebas. La guía provee plantillas y ejemplos que apoyan la aplicación de las mismas.

3.3.6 Actividad MoProSoft: Creación de los casos de prueba (no incluida en MoProSoft)

Considerando que esta actividad es esencial para poder aplicar cualquier tipo de prueba, GPS ofrece los elementos para ejecutarla.

Primero se deben "definir y crear casos y escenarios de prueba". Los casos de prueba sirven para documentar la ejecución de las mismas. El diseño de los casos depende del nivel en el cual se apliquen las pruebas: de integración, de sistema o de aceptación. GPS proporciona información para realizar cada una de ellas. También se señalan los datos mínimos que debe contener todo caso de prueba, dependiendo del grado de madurez del área de pruebas de la empresa. Por otra parte, el contenido de los casos de prueba puede variar según la importancia del requisito y condiciones a probar, y del tiempo estimado para desarrollar los casos de prueba. GPS ofrece modelos de diferentes tipos de casos de prueba que pueden ayudar a que cada organización defina los propios ajustándolos a sus necesidades específicas. Un ejemplo de caso de prueba sencillo se presenta en la tabla 11.

Id. Req. GN.0003	**CP. 0001.func**.: verificar menú de administrador al entrar al sistema	**Fecha aprobación**: 12/08/2010
Versión: 001	**Objetivo**: Verificar que al entrar al sistema con un usuario de tipo "administrador" se muestre el menú de "administrador".	
Paso:	**Acción**:	**Resultado esperado**:
1	Ingrese al sistema con un usuario de administrador.	Luego del "login" correcto, la pantalla muestra el menú general y el submenú "administrador" con todas las opciones habilitadas.

Tabla 11. Ejemplo de Caso de Prueba Simple

Además de los casos de pruebas se necesita la "creación de datos de prueba". La obtención de los datos con los cuales realizar las pruebas es un paso esencial, de ello depende en gran medida el éxito de las mismas. En general, es responsabilidad del experto de negocio obtener/generar los datos de pruebas. Dependiendo del tipo de prueba se elegirán los datos apropiados. Los datos también deben administrarse, y GPS explica cómo hacerlo en cada caso.

3.3.7 Actividad MoProSoft: Realización de la fase de Integración y Pruebas (A5)

GPS complementa esta actividad con otras tres (ver tabla 5) que permiten incluir las pruebas durante la ejecución de la misma. A continuación se describen brevemente cada una de ellas.

La actividad "realizar integración y pruebas" se divide en etapas. En la primera de ellas (crear el plan de ejecución de las pruebas de integración) se genera un plan que contiene el detalle de las actividades y procedimientos a seguir durante la ejecución de las pruebas. En el plan, los casos de prueba están agrupados en bloques de acuerdo a su objetivo (o funcionalidad) y a su vez ordenados según una secuencia establecida. GPS da lineamientos para formar los bloques de ejecución, así como para generar el plan de pruebas. En la figura 3 se presenta un ejemplo.

	Plan de Ejecución	**Pruebas de Sistema**
	Módulo / Funcionalidad	**Caso de Prueba**
	Ambiente	Actividad: Instalar el ambiente de ejecución.
	Ambiente	Actividad: Cargar datos de configuración.
	Instalación	Instalación de sistema en equipo de pruebas.
A	General (GN)	CP.0001.func: Verificar pantalla de inicio.
		CP.0002.func: Verificar menú de administrador al ingresar al sistema.
		CP.0003.func: Verificar menú de Ventas al ingresar al sistema.
B		CP.0004.func: …
		CP.0005.func: …
C	Ventas (VT)	CP.0120.func: Crear una venta a crédito.
		CP.0121.func: Cancelar una venta a crédito.
		CP.0122.func: …
	Ambiente	Ejecución de inicio de día laboral.
	Ambiente	Ejecución de cierre de día laboral.
D	Compras (CM)	CP.0030.func: Verificar botón de exportar una cotización.
		CP.0031.func: …
		CP.0032.func: …
		CP.0033. func: …

Figura 3. Ejemplo de bloques de ejecución de pruebas

La segunda etapa (ejecutar pruebas de humo) está orientada a comprobar que un cierto módulo o componente trabaja de la manera esperada antes de empezar con la ejecución de los casos de prueba formales del sistema. Por último, realizar pruebas de integración, hace referencia a ejecutar las pruebas de integración a medida que los componentes del sistema se van integrando. Es importante que se registren las evidencias obtenidas, por lo que GPS ofrece una lista de indicadores que deben considerarse así como un modelo de cómo realizar esta tarea. Adicionalmente, GPS propone la elaboración de un reporte en el que se debe incluir el número de veces que se probó cada pieza de software y el tiempo ocupado para la solución de defectos. Los datos que aporta este reporte permiten realizar proyecciones sobre el plazo de entrega y mejorar el manejo de los riesgos del proyecto. También sirve a la gerencia para evaluar la calidad del producto luego de esta fase y, en consecuencia, dar por terminada o no la misma.

La actividad "realizar las pruebas de sistema" comprende dos tareas: crear el Plan de Ejecución de las Pruebas de Sistema y realizar las pruebas de sistema. En la primera se genera el plan que contiene todas las actividades a ejecutar en esta fase, incluyendo aquellas que son complementarias como la configuración del ambiente. Es importante que en el plan se considere el tiempo requerido para la corrección de los defectos. GPS proporciona un modelo para elaborar este plan de pruebas. La segunda consiste en la ejecución de las pruebas según el plan establecido. Por la relevancia que tiene en el éxito del proyecto, GPS propone dividir esta actividad en ocho pasos, cada uno de ellos más fácil de entender y de aplicar. Asimismo, proporciona algunos elementos requeridos, como formatos para registro de errores (ver figura 4) y administración de defectos entre otros.

# Defecto	**Fecha Alta**	**Caso de Prueba**	**Revisor (*Tester*)**	**Severidad**	**Paso**	**Fecha Cierre**
Def.0001	10-Ago-2010	CP.0001.func	Alain Ponce	2- Alta	6	
Defecto: Error en el alta de un nuevo cliente. **Descripción**: Al dar de alta a un nuevo cliente el sistema despliega el mensaje "Necesita ingresar al menos un número de teléfono". Sin embargo el registro ya tiene cargados 2 números en los campos correspondientes. **Consideraciones generales**: Los dos números telefónicos dados son de celulares, sin embargo no existen restricciones sobre el tipo de teléfono.						

Figura 4. Ejemplo de Registro de Defectos

Por último, "realizar las pruebas de aceptación" (no incluida en MoProSoft). Su objetivo es validar el sistema como un todo, desde la perspectiva del negocio. Para enfatizar su importancia y la diferencia entre éstas y las pruebas de sistema se presentan como una actividad aparte. Estas pruebas también deben seguir el plan y registrar y administrar los defectos a lo largo de su ciclo de vida.

3.3.8 Actividad MoProSoft: Realización de la fase de Cierre (A6)

GPS complementa esta actividad con otras dos (ver tabla 5) que permiten incluir las pruebas durante la ejecución de la misma.

La actividad "decisión de finalización del proyecto" (no incluida en MoProSoft) es de fundamental importancia. En ella se analiza la información recolectada a lo largo de todas las pruebas y, con base en ella, se decide si se acepta o no el proyecto. La organización puede definir un límite máximo de pruebas fallidas aceptables. GPS presenta una escala obtenida a partir de la experiencia en la industria del software mexicana (ver tabla 12).

Fase de pruebas	**Tipo de efecto de la aplicación en el negocio**			
	Secundaria	**Importante**	**Fundamental**	**Ventaja competitiva**
Unitarias	N/A	N/A	30 %	30 %
De integración	35 %	25 %	10 – 15 %	30 %
De sistema	30 %	20 %	5 – 10 %	10 %
De aceptación	30 %	20 %	2 – 5 %	1 %

Tabla 12. Pruebas fallidas "aceptables"

La actividad **manual de mantenimiento** permite crear un documento de rastreo de todos los módulos del sistema con las pruebas que fueron ejecutadas sobre ellos. El documento describe la configuración del software y los ambientes usados para el desarrollo y para las pruebas, brindando así información básica para el mantenimiento de cada módulo. GPS define claramente los elementos que debe contener el manual.

4. Conclusiones

La guía presentada en este trabajo resulta de la experiencia en el desarrollo de sistemas en PYMES mexicanas y de la necesidad que tienen organizaciones de este tipo de formalizar

sus procesos. MoProSoft es un modelo de procesos pensado y diseñado para PYMES; sin embargo, al igual que otros modelos similares, no proporciona todas las herramientas requeridas para la implementación del modelo dentro de la organización. Adicionalmente, este tipo de empresas generalmente no cuenta con los recursos suficientes para desarrollar esos elementos. Esta guía complementa a MoProSoft en el área de pruebas, definiendo para cada una de las categorías del modelo las actividades de pruebas mínimas necesarias para cuidar la calidad del producto. Se incluyeron además algunas actividades de pruebas no contempladas en el modelo, debido a que son de fundamental importancia para aumentar la competitividad y permanencia en el mercado de este tipo de empresas. Para facilitar la aplicación de las pruebas, GPS provee plantillas y ejemplos que pueden ser usados como base para definir los documentos propios en cada organización.

GPS también proporciona valores estándares de indicadores de pruebas en la industria del software que, para las PYMES que no están dedicadas a crear o mantener software, pueden ser de ayuda para establecer contratos de negocio con desarrolladores externos. Mientras que, a las PYMES dedicadas al desarrollo y mantenimiento de software que no cuenten con datos históricos, les permitirán hacer inicialmente sus estimaciones de recursos.

Por último, es importante destacar que la guía presenta y explica conceptos básicos relacionados a las pruebas de software, los cuales pueden ser usados por miembros de las PYMES como medio de capacitación en dicha disciplina.

Agradecimientos

Se agradece el apoyo de la Asociación Mexicana de Cultura A.C.

Referencias

[1] Software Engineering Institute, *Capability Maturity Model Integrated (CMMI).* CMU/SEI-2006-TR-008, Carnegie Mellon University, 2006.

[2] ISO/IEC International Standard Organization/International Electrotechnical Commission, *ISO/IEC 12207 Systems and software engineering —Software life cycle processes*. ISO/IEC, 2008.

[3] ISO/IEC International Standard Organization/International Electrotechnical Commission, *ISO/IEC 15504 Software engineering — Process assessment*. ISO/IEC, 2004.

[4] Modelo de Procesos para la Industria de Software (MoProSoft). Comunidad MoProSoft. www.comunidadmoprosoft.org.mx/COMUNIDAD_MOPROSOFTADM/Documentos/V_1.3_MoProSoft_por_niveles_de_capacidad_de_procesos.pdf [Consultado: diciembre 2009].

[5] Anacleto, A., Von Wangenheim, C., Salviano, F. y Savi, R., "Experiences gained from applying ISO/IEC 15504 to small software companies in Brazil". En: *Proceedings of the Fourth International SPICE Conference on Process Assessment and Improvement. Lisboa (Portugal), 27 al 29 de abril de 2004*, pp.33-37, 2004.

[6] Laporte, C., *ISO Working Group Mandated to Develop Standards and Guides for Very Small Entities involved in the Development or Maintenance of Systems and/or Software,* http://profs.logti.etsmtl.ca/claporte/English/VSE/. [Consultado: marzo 2010].

[7] Laporte, C., April, A. y Renault, A., "Applying ISO/IEC Software Engineering Standards in Small Settings: Historical Perspectives and Initial Achievements". En: *Proceedings of the Six International SPICE Conference on Process Assessment and Improvement, Luxembourg, 4 al 5 de mayo de 2006*, pp. 57-62, 2006.

[8] Laporte, C., Alexandre, S. y Renault, A. "The Application of International Software Engineering Standards in Very Small Enterprises". SQP, vol. 10, nº 3, pp 4-11, 2008.

[9] Oktaba, H. "MoProSoft: A Software Process Model for Small Enterprises". En: *Proceedings of 1st Int'l Research Workshop for Process Improvement in Small Settings. Special report CMU/SEI-2006-SR-001, Software Engineering Institute*, pp. 93-101, 2006.

[10] Myers, G. *The Art of Software Testing*, John Wiley & Son, 2004.

[11] Guardati, S. y Ponce, A. *Guía de Pruebas de Software (GPS) para MoProSoft*. Comunidad MoProSoft. 2010. Disponible en: http://comunidadmoprosoft.org.mx/Aportaciones/GPS_para_MoProSoftv1.0.pdf

[12] Amengual E., Mas A. y Mesquida A., "Team SPICE: A SPICE-Based Teamwork Assessment Model", *Communications in Computer and Information Science*, vol. 99, pp. 37-47, 2010.

REICIS

ACTUALIDAD INVITADA

Aseguramiento de calidad del software en Administraciones Públicas

Marcos Blanco Galán
Coordinador del Centro de Certificación y Calidad del Software en CESJE

Introducción

La capacidad de innovar y ofrecer valor añadido sobre servicios y productos se ha convertido en una necesidad para garantizar la supervivencia y el desarrollo óptimo de las organizaciones. Si nos centramos en el sector de las TIC (Tecnologías de la Información y las Comunicaciones), observamos que infraestructuras y soluciones de base tecnológica integran a día de hoy el soporte crítico de muchos servicios públicos. Este hecho, evidenciado claramente en los últimos años, se reafirma con la Ley 11/2007, de 22 de junio, sobre el acceso electrónico de los ciudadanos a los servicios públicos. Los ciudadanos, como contribuyentes y principales beneficiarios, están en su derecho de exigir servicios de calidad basados en estas infraestructuras tecnológicas. Cuando hablemos de Calidad del Software nos plantearemos las siguientes cuestiones:

- ¿Se satisfacen realmente las necesidades del cliente?
- ¿Qué características internas del producto pueden utilizarse para medir la calidad?
- ¿Qué valor recibe el cliente en términos de utilidad y garantía?

La Junta de Extremadura, consciente de la complejidad del escenario y conocedora de los problemas comunes existentes en los proyectos de desarrollo software, artículo en 2009 la creación de un Centro de Excelencia de Software que, entre otras funciones, serviría para velar por la calidad de las soluciones informáticas adquiridas desde la Administración en procedimientos de licitación. Actualmente, el Centro de Excelencia de Software José de Espronceda (CESJE), a través del Centro de Certificación y Calidad del Software (CESJE-

CQS), continúa ofreciendo sus servicios y desarrollando nuevos proyectos en el ámbito de la Calidad del Software.

Consideraciones sobre la industria del desarrollo de software

El mercado de las TIC se caracteriza por estar en continua evolución. Sin embargo, sigue existiendo un amplio desconocimiento de las metodologías formales y estrategias dirigidas a mejorar la calidad final de los productos de software y de los procesos que intervienen en su creación.

Técnicamente hablando, ya no es suficiente con que una aplicación sea funcionalmente completa acorde a su propósito. Ésta también debe satisfacer otros aspectos en términos de seguridad, rendimiento, accesibilidad, mantenibilidad, etc. Estos requisitos, denominados no-funcionales en el argot de la Ingeniería del Software, no siempre vienen definidos formalmente. Sin embargo, constituyen propiedades inherentes del producto que, de una manera u otra, son percibidos y muy valorados por los clientes y usuarios finales.

Cabe destacar también una tendencia hacia proyectos comúnmente denominados de "alto riesgo". La rápida evolución del negocio y los requisitos cambiantes impiden abordar etapas largas en el proceso de desarrollo. En este escenario, las metodologías ágiles parecen ganar terreno sobre metodologías más pesadas de modelo en cascada. No obstante, con independencia de la metodología finalmente aplicada, siempre deberán considerarse en el proceso actividades de aseguramiento de la calidad.

Por otra parte, no todas las organizaciones pueden asumir los costes derivados de un equipo o departamento destinado exclusivamente a validar las aplicaciones desarrolladas. Como tendencia ampliamente generalizada, las pruebas son acometidas de forma íntegra por el propio equipo de desarrollo. En este sentido, el hecho de haber participado en las etapas de diseño e implementación, favorece la adopción de una visión excesivamente técnica y limitada a la hora de diseñar y ejecutar los planes de pruebas.

Como consecuencia, muchas empresas terminan sacrificando la calidad del software (consciente o inconscientemente) a cambio de reducir tiempos de entrega, recursos y costes asociados, ya sea por motivos de escasez, estratégicos o cultura de la organización. El resultado final es la obtención de productos que no siempre satisfacen las necesidades y expectativas de los clientes y usuarios.

El contexto de las Administraciones Públicas

Según recoge el informe de INTECO "Estudio sobre la certificación de la calidad como medio para impulsar la industria del desarrollo de software en España", las nuevas exigencias de calidad suponen una de las principales causas del incremento de la competitividad, debiendo ser la propia Administración Pública quien impulse la adopción de modelos de calidad en la empresa española.

En España ya son varias las entidades públicas que confían en modelos de calidad reconocidos en la industria del desarrollo de software, tales como ISO-15504/SPICE o CMMI. Sin embargo, también es cierto que no todas las organizaciones, bien por su naturaleza y/o recursos limitados, pueden permitirse implantar estos modelos y obtener las certificaciones asociadas. Esto supone una importante barrera a la hora de optar a concursos, lo cual no implica necesariamente que los productos generados por estas empresas no sean válidos.

En este escenario complejo plagado de matices, donde conviven Administración Pública, empresas desarrolladoras de software y clientes/usuarios, resulta extremadamente difícil establecer un punto de equilibrio en el que, teniendo presente la capacidad y recursos limitados de muchas empresas tecnológicas, sea posible garantizar que los usuarios finales acaban recibiendo los servicios con el nivel de calidad esperado.

Un nuevo marco de actuación dentro de las Administraciones Públicas

Como factores claves del éxito se requieren actividades de verificación y validación imparciales llevadas a cabo por personal cualificado, garantizando en todo momento la independencia operativa y la autoridad respecto a los responsables del desarrollo. Con el propósito de asegurar la calidad de las soluciones informáticas recibidas por las Administraciones Públicas, parece adecuado establecer un marco de actuación consistente en la externalización de un servicio de testing o pruebas de software. Centrado principalmente en la gestión de la calidad a nivel de producto, este servicio se integraría en el ciclo de vida del software con independencia de la metodología utilizada por el proveedor del desarrollo.

Siguiendo esta línea de trabajo, CESJE define y pone a disposición de las entidades públicas interesadas el servicio 'Pruebas de Software'. Entre las soluciones informáticas

evaluadas hasta el momento destacan las aplicaciones que integran la plataforma JEXEELL, destinada a proveer un entorno basado en Software Libre para el desarrollo de los procesos de e-Administración en los ayuntamientos de la región.

En marzo del presente año, en colaboración con la empresa Optimyth Software, CESJE-CQS tuvo la oportunidad de compartir su experiencia en la materia suscitando el interés de representantes procedentes de diferentes entidades públicas, entre las que se encontraban el Ministerio del Interior, Ministerio de Industria, Ministerio de Economía y Hacienda, Ministerio de Defensa y Junta de Castilla y León.

Sin lugar a dudas, con el diseño y puesta en funcionamiento de este nuevo marco de actuación, podemos afirmar que Extremadura se erige como pionera y referente en la provisión de servicios de Aseguramiento de la Calidad aplicados al modelo de software regional dentro de la Administración Pública.

Perfil profesional

Marcos Blanco Galán es Ingeniero en Informática e Ingeniero Técnico en Informática de Sistemas por la Universidad de Extremadura. Desde 2009 ejerce como Coordinador del Centro de Certificación y Calidad del Software en el Centro de Excelencia de Software José de Espronceda (CESJE). Anteriormente fue Jefe de Proyectos de la Cátedra Telefónica de la Universidad de Extremadura y Jefe de Proyectos del Centro Internacional de Referencia Linux promovido por IBM. Doctorando en Tecnologías Informáticas, es miembro del Comité Extremeño de itSMF España y coordinador de formación TIC en Balbo, Consultoría y Formación, S.L.