Revista Española de Innovación, Calidad e Ingeniería del Software

REICIS

Volumen 4, No. 3, octubre, 2008

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

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

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

Contenidos

REICIS

El tamaño sí importa en la mejora de procesos

Tanja E.J. Vos, Jorge Sánchez Sánchez, Maximiliano Mannise
SQUaC (*Software Quality, Usability and Certification*)
Instituto Tecnológico de Informática
Universidad Politécnica de Valencia,
Camino de Vera s/n - 46022 Valencia, Spain
{tanja,jordisan,mmannise}@iti.upv.es

Resumen: A partir de las experiencias en la aplicación de procesos de mejora de testeo en diferentes PYME, detectamos que las metodologías más habituales de ese tipo están orientadas a organizaciones mucho más grandes y son, por tanto, difícilmente aplicables a ese perfil de empresas, con recursos limitados y poca madurez en los procesos de testeo. Proponemos, por tanto, una serie de acciones sencillas y concretas que las PYME pueden realizar sin dedicar muchos recursos, obteniendo resultados rápidamente y preparándolas para un posterior proceso de mejora más formal.

Palabras clave: Testeo, software, PYME, mejora, procesos

Abstract: Based on experiences in implementing testing processes improvement in different SMEs, we found that the most common methodologies are aimed at much larger organizations and are, therefore, difficult to implement in a company profile that has limited resources and low maturity in the testing processes. We therefore propose a series of simple and concrete actions that SMEs can do to spend little resources and get fast results, while getting prepared for further, more formal improvement processes.

Keywords: software, testing, SME, process, improvement

1. Introducción

Este artículo describe nuestras experiencias durante los proyectos de transferencia de tecnología y servicio de consultoría con objeto de ayudar a las empresas a resolver un "problema con el testeo" detectado por ellas, o bien únicamente para apoyarlas en la mejora de sus prácticas en testeo de software. En los siguientes apartados describiremos el entorno y las características de dichos proyectos.

1.1. El ITI (Instituto Tecnológico de Informática)

El ITI es un instituto de investigación sin ánimo de lucro situado en las instalaciones de la Universidad Politécnica de Valencia (UPV), cuyo principal objetivo es aplicar los

conocimientos obtenidos de la investigación e innovación científica en tecnología punta a las empresa de pequeño y mediano tamaño (PYME) relacionadas con tecnologías de información y comunicación (TIC).

El instituto consta de siete grupos de investigación, muchos de ellos liderados por profesores universitarios. Las experiencias descritas en este artículo provienen del grupo SQUaC, que trabaja en las áreas de calidad, usabilidad y certificación de software.

La misión del ITI incluye, como uno de sus principales objetivos, construir puentes entre el conocimiento y la tecnología obtenidos de la investigación científica, por un lado, y la necesidad de soluciones simples, prácticas y robustas para la industria relacionada con las TIC, por otro. Para conseguir este objetivo, el ITI apoya a las empresas con formación a medida, transferencia de tecnología, soluciones estándar propias, y proyectos de consultoría en la propia empresa.

1.2. El ITI (Instituto Tecnológico de Informática)

Muchas de las empresas a las que apoyamos son pequeñas compañías de software en Valencia, España. Estas empresas se caracterizan por lo siguiente:

- Venden un producto software personalizado para determinados usuarios finales.
- El software, inicialmente programado únicamente por uno o dos desarrolladores, ha crecido desde algo pequeño, transparente y fácil de mantener hasta ser un monstruo enorme, opaco e imposible de mantener por un mayor grupo de desarrolladores.
- Sus clientes están malacostumbrados; primero, porque se les permite proporcionar requerimientos muy vagos para los nuevos desarrollos. Segundo, porque están acostumbrados a que todas sus demandas sean implementadas. Esto lleva frecuentemente a que existan múltiples versiones del mismo software.
- No existen documentos de requerimientos escritos, o los que existen están poco estructurados, son poco detallados o incompletos.
- No existen procesos de testeo estructurados ni personal específicamente dedicado al testeo. De hecho, en muchas ocasiones ni siquiera se realizan actividades específicas de testeo.

- Existe escasa información y gestión de los defectos en el software. La que existe está muy centrada en las incidencias notificadas por clientes.

1.3. Sus proyectos (y sus problemas)

En muchas ocasiones las empresas acuden a nosotros debido a que tiene problemas que creen que pueden resolverse mediante testeo. Los problemas son muchas veces los mismos:

- Los clientes se quejan porque encuentran muchos errores, o porque las funcionalidades de las aplicaciones no se corresponden con sus necesidades.
- No se cumplen los plazos de entrega y presupuesto acordados. En muchos casos se sacrifica el testeo de software por urgencias del proyecto.
- No se gestionan adecuadamente los defectos encontrados durante las pruebas internas, ni tampoco los reportados por los clientes.
- Se dedican demasiadas horas a corregir defectos y/o malentendidos en los requerimientos.
- Las múltiples versiones del software son difíciles de gestionar y/o testear.
- Existe poca o ninguna información sobre la calidad de los desarrollos.

1.4. El testeo de software y sus niveles

Para cualquier compañía (también para las PYME) que se dedique al desarrollo de software, sea cual sea la metodología que utilice, el testeo debe ser una parte fundamental tanto para asegurarse de que los productos están siendo desarrollados correctamente (verificación) como de que los productos satisfarán los requisitos del usuario (validación).

En cualquier caso, un testeo adecuado durante el ciclo de desarrollo cumple que ([1]):

- El testeo se inicia en las primeras etapas del ciclo de vida del software
- No sólo se testea el software final, sino también los requerimientos, manuales, otros documentos, etc.
- Los testeadores están involucrados en la captura de requerimientos

Los niveles típicos de testeo durante el ciclo de vida de una aplicación son:

- Testeo unitario (o de componentes). El código fuente del software suele estar dividido en unidades más o menos aisladas, también llamadas programas, módulos, componentes o clases. El testeo unitario pretende asegurar que cada una

de esas unidades cumple su especificación por separado. Este testeo es habitualmente realizado por el mismo desarrollador que escribe el código.

- Testeo de integración. Una vez que las unidades han sido escritas y testeadas, la siguiente etapa consiste en unirlas para crear el sistema completo; esto es llamado integración. El propósito del testeo de integración es descubrir defectos en las interfaces y en las interacciones entre diferentes componentes o sistemas.
- Testeo de sistema. Una vez que se ha comprobado que los componentes funcionan y se integran correctamente entre ellos, el siguiente paso es considerar la funcionalidad completa del sistema como un todo; a esto se le llama testeo de sistema. Este testeo es necesario ya que los niveles inferiores de testeo (unitario y de integración) utilizan una visión parcial y no son representativos de las condiciones reales de funcionamiento del sistema. El testeo de sistema, usualmente realizado por un equipo ajeno a su desarrollo, puede referirse tanto a requerimientos funcionales (comprobando que el sistema realiza las funciones deseadas) como a requerimientos no funcionales, como son: rendimiento, carga, fiabilidad, usabilidad, etc.
- Testeo de aceptación. El propósito del testeo de aceptación es dar confianza al usuario final de que el sistema funcionará de acuerdo a sus expectativas. La base para el testeo de aceptación es el documento de especificación de requerimientos, comprobando en la práctica que el sistema los cumple adecuadamente. Este testeo suele ser totalmente independiente del resto de niveles de testeo y de la implementación del sistema, y puede ser llevado a cabo por los propios usuarios finales del sistema (con la posible participación de miembros del equipo de desarrollo), en las instalaciones de desarrollo o en las del propio cliente.

2. La mejora de procesos y las PYME

2.1. Metodologías de mejora de procesos

Existen bastantes metodologías y técnicas para la mejora de procesos en general, y para la mejora de procesos de testeo en particular (algunas de ellas mencionadas en [2] y [3]).

Metodologías como SPICE [4], desarrollada por ISO/IEC, pretenden ser un estándar para la evaluación de procesos software en general. Otras, como CMMI [5], se orientan de manera todavía más global, abarcando todo el desarrollo y mantenimiento del producto, e integrando diferentes áreas de conocimiento que antes eran tratadas por separado, como ingeniería de software, ingeniería de sistemas o aprovisionamiento.

Otras autores, ante la insuficiente atención que prestan esos modelos globales al testeo de software, han desarrollado metodologías de mejora específicas como Test Organization Maturity (http://www.gerrardconsulting.com/default.asp?page=/tomoverview.html), Test Improvement Model (TIM) [6], Testing Madurity Model (TMM) [7] o Test Process Improvement (TPI) [8].

2.2. Mejora de procesos en PYME, en la práctica

El mayor problema con los modelos anteriores es que fueron diseñados inicialmente para organizaciones más grandes, y no son fácilmente adaptables a estructuras mucho más pequeñas, como nuestras PYME. El primer motivo es que el coste y la duración de un proceso de evaluación son desproporcionados respecto a los recursos disponibles en una PYME. Además, su nivel de madurez, especialmente en lo que respecta a testeo de software, es normalmente muy bajo; por ejemplo, en una evaluación inicial rápida, la mayoría de las PYME con las que trabajamos únicamente alcanzan el nivel mínimo de madurez de TPI (el nivel A) en el área de entorno de oficina.

Varios estudios (por ejemplo [10]) muestran que un gran número de proyectos de mejora de procesos en PYME basados en modelos como CMMI encuentran problemas graves, y que un importante porcentaje de esos problemas (53%) está relacionado con el tamaño de la empresa; el éxito de un proyecto de mejora de procesos crece con el número de personas a cargo de procesos de software. Como se indica en [11], en este tipo de proyectos y con empresas de ese tamaño, es imposible recoger suficiente material como para realizar un análisis estadístico sólido, siendo en estos casos mucho más importante el factor personal, involucrando desde el principio a las personas clave (las que poseen los conocimientos).

Por otro lado ([12]), la rigidez y el formalismo de esas metodologías conllevan para las PYME un incremento de la burocracia que se convierte en un lastre ante la flexibilidad y la innovación que dichas empresas necesitan en un entorno dinámico sobre el que

prácticamente no tienen control. Reducir el rigor de las metodologías no basta para reducir esos inconvenientes.

Un factor clave en el éxito o fracaso de este tipo de proyectos reside en la motivación que impulsa a la PYME a emprender ese proceso de mejora ([13]). Para algunas se trata de un "mal necesario" que deben afrontar por la presión de clientes importantes como organismos públicos o grandes empresas; en estos casos el proyecto se considera simplemente como un coste más, sin auténticas expectativas de mejora. Por otro lado, cuando los responsables de la empresa ven en el proyecto de mejora una verdadera oportunidad de optimizar en la práctica sus procesos y sistemas internos, aumentan las posibilidades de obtener resultados positivos.

En el caso concreto de España, estudios como [9], además de resaltar la especial importancia de las PYME (representando más de un 99% de las empresas en el sector software), ponen de manifiesto que metodologías como CMMI y SPICE son bastante conocidas, gracias en gran medida a las subvenciones de organismos públicos para su adopción. Sin embargo, parece que el interés de las empresas está más centrado en la certificación frente a terceros que en la mejora interna de procesos: existe conocimiento sobre la existencia de los estándares y su utilización fuera de España, pero no se conocen los detalles sobre su uso interno y todo aquello que implica su adopción a la hora de aplicarlas en la práctica. Como resultado, el nivel de adopción es muy bajo, siendo las barreras que esgrimen las PYME los altos costes de la adopción del modelo, el cambio en los métodos de trabajo que implican, la necesidad de realizar un cambio organizativo o la dificultad de evaluar los beneficios de implantarlas.

En resumen, las experiencias de las PYME con las metodologías de mejora de procesos son muchas veces problemáticas, debido a la inversión excesiva en tiempo, en herramientas, etc., que les supone aplicar modelos que están pensados para organizaciones más grandes, con más recursos y mayor madurez en esos procedimientos. Esas dificultades se convierten en un abismo que separa a la PYME de los procesos de mejora (Figura 1)..

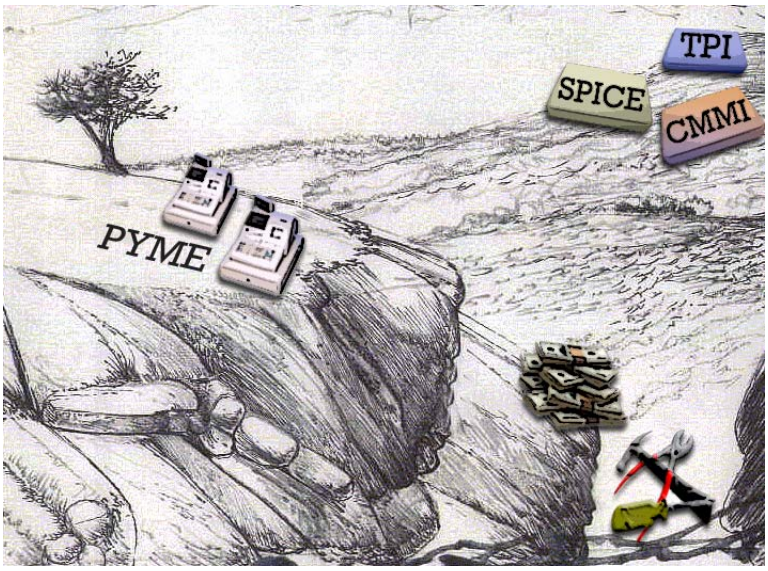

Figura 1. En el abismo entre las PYME y las metodologías de mejora se pierden muchos recursos: dinero gastado, herramientas abandonadas, etc..

3. Preparando los modelos de mejora para las PYME

Ante todos esos problemas que surgen a la hora de aplicar los modelos de mejora a las PYME, se han realizado diversos intentos de adaptar o crear nuevos modelos específicamente orientados a las características de empresas de pequeño tamaño.

En general, consisten en simplificaciones y/o adaptaciones de modelos más complejos como CMMI (que es usado como referencia en IT Mark[1], especialmente implantada en España) o SPICE/ISO 15504 (que es tomado como base de Rapid Assessment for Process Improvement for software Development, RAPID [15]); o en la combinación de varios de ellos (como hace [16]).

Otros modelos, como Minimal Test Practice Framework (MTPF) [17], están específicamente centrados en las actividades de testeo de software e intentan adaptarse a la naturaleza concreta de las PYME: escasez de recursos, resistencia al cambio interno, flexibilidad, etc.

En España según [9], ITmark parece tener especial importancia, aunque existen también iniciativas como el programa SoftAragón, que ha desarrollado un modelo CMMI

[1] http://www.esi.es/index.php?op=15.1.2

reducido para su aplicación a las PYME aragonesas, o MESOPYME [18], especialmente orientado a la fase de implementación de la mejoras.

Todas estas adaptaciones hacen que los modelos de mejora (y, en concreto, los de mejora del testeo) sean más fácilmente aplicables a las PYME; pero esas metodologías específicas pueden no ser suficientes, sobre todo debido a la poca madurez y conciencia que encontramos en la práctica en las empresas. ¿No hay nada que podamos hacer en la propia PYME para facilitar la implementación de una metodología de mejora de procesos de testeo? Nosotros creemos que sí.

4. Preparando a las PYME para la mejora de procesos de testeo

Antes de empezar a aplicar alguno de los modelos de mejora vistos anteriormente (tanto generales como adaptados a pequeñas organizaciones) en una PYME con las características que veíamos al principio (ausencia de procesos de testeo estructurados, sin personal dedicado, etc.), proponemos preparar a la empresa con dos acciones muy prácticas y sencillas:

- Testeo unitario por programadores
- Personal exclusivamente dedicado al testeo

4.1 Testeo unitario por programadores

La única responsabilidad de los programadores en cuanto al testeo sería realizar algún tipo de testeo unitario sobre sus desarrollos, dejando a su elección qué y cómo hacerlo. Confiemos en el conocimiento de los programadores: saben lo que quieren programar, así que dejemos que sean ellos los que comprueben si lo hacen correctamente.

Para tener éxito en este punto sería necesario:

- Obligar a los programadores a que realicen testeo unitario. Al fin y al cabo, la tarea de los programadores no es simplemente producir programas, sino producir código de calidad, libre de errores; y una de las maneras de conseguirlo es mediante el testeo.
- Motivar a los programadores para que realicen el testeo. Para ello pueden utilizarse diferentes técnicas. Por ejemplo, hacer reuniones periódicas específicamente dedicadas al testeo (“cada viernes por la mañana”); realizar seminarios informales

para exponer los avances conseguidos ("mirad lo que he conseguido/encontrado esta semana"); o utilizar otros medios para compartir esa información, como Intranets [12] o incluso el "testeo en el aseo" que realiza Google (http://googletesting.blogspot.com/2007/01/introducing-testing-on-toilet.html).

- Darles libertad para realizar el testeo según crean conveniente. De hecho, los programadores se enfrentan continuamente con nuevas tecnologías y técnicas que tienen que utilizar para resolver algún tipo de problema; constantemente buscan, aprenden y aplican esas nuevas tecnologías. Así que ¿por qué no van a hacerlo con los frameworks de testeo unitario y otras herramientas existentes?
- Proporcionarles los recursos necesarios. Sobre todo necesitarán tiempo asignado específicamente al testeo, tanto para aprender e investigar los mejores métodos como para realizar efectivamente el testeo unitario. Por su trabajo, normalmente ya dispondrán del entorno (hardware, instalaciones, etc.) requerido para el testeo, pero si no se planifica el esfuerzo extra de realizar el testeo unitario, lo más probable es que no le dediquen el tiempo necesario.

Para complementar sus conocimientos, podría ser necesario algún tipo de formación orientada a explicar en términos generales qué es testeo unitario, así como una visión global de las técnicas y herramientas existentes. Todo ello sin entrar en detalles, dejando que sean los programadores los que elijan cómo aplicar en la práctica esos conocimientos a su entorno de trabajo.

4.2. Personal exclusivamente dedicado al testeo

Consistiría en contratar al menos 1 persona dedicada exclusivamente al testeo de alto nivel. Asignándole como único objetivo el encontrar errores, además de encontrarlos en la práctica, sacará a la luz necesidades para realizar correctamente su trabajo (como mejor definición de requerimientos, gestión de defectos, análisis de riesgos, planificación, etc.) que, además mejorarán los procesos de desarrollo de la empresa.

Idealmente, el responsable de esa tarea sería alguien con conocimientos y experiencia previa en testeo de software. No obstante, teniendo en cuenta el mercado laboral actual y la dificultad de encontrar y contratar profesionales con esos requisitos (especialmente para las PYME), podemos aceptar que se trate de una persona sin experiencia previa en testeo. En general, se trataría de alguien con perfil técnico (por

ejemplo, Ingeniero en Informática) y ciertas habilidades en otros campos: comunicación, planificación de tareas, etc.; en esta situación será muy importante su capacidad de autoformación. En todo caso, recordemos que estas sugerencias pretenden conducir, no a la estructura de testeo definitiva dentro la organización, sino a una preparación básica previa antes de acometer un proceso de mejora más formal.

La persona asignada a esa tarea podría ser alguien que ya pertenece a la empresa, aunque en la práctica suele ser muy difícil que una persona que ya tiene responsabilidades asignadas abandone todas sus tareas anteriores para dedicarse únicamente al testeo. Contratando nuevo personal se evita ese problema, lo que compensa otros inconvenientes, como el aumento de costes o el desconocimiento que alguien venido de fuera pueda tener de la organización.

4.3. Otros aspectos

Además de esos dos puntos, pueden ser de utilidad algunos cursos básicos sobre testeo para incrementar la conciencia de su necesidad en la empresa, aunque no demasiados para que la PYME no se vea abrumada por la cantidad de cosas que NO están haciendo.

Tenemos que estar preparados para que la PYME sea en ocasiones reacia a aceptar esa libertad (“que hagan lo que crean necesario”) al principio. Sin embargo, es fácil explicarles que, si partimos de una situación en la que prácticamente nada se está haciendo, cualquier paso es ya un gran avance en el camino hacia la mejora de procesos. Es importante que la PYME entienda que, en este primer momento, el factor más importante es la motivación.

Así, tras algún tiempo (proponemos unos 6 meses), la empresa habrá adquirido ciertos conocimientos y experiencias en testeo, y podremos empezar con acciones más ambiciosas: aplicar una metodología más formal como TPI, cursos más específicos en función de las necesidades de la PYME, etc.

5. Conclusiones

Con las dos sugerencias que ofrecemos, y basándonos en nuestra experiencia, creemos que una PYME se encontrará en poco tiempo en una situación mucho más propicia para poder acometer con éxito un proceso más ambicioso de mejora del testeo, como pueda ser aplicar

una metodología de mejora más formal, aumentar los recursos dedicados al testeo, etc. En todo caso, suponen una inversión relativamente pequeña y cualquier avance en la madurez de la empresa en temas de testeo que se consiga con estas prácticas supondrá un avance importante respecto a la situación de partida que nos encontramos habitualmente.

Algunas de las PYME con las que hemos trabajado están empezando a aplicar (de modo parcial) estas sencillas sugerencias, obteniendo de inicio resultados bastante positivos. Nos queda pendiente estudiar y reflejar las experiencias y los resultados obtenidos en la práctica después de transcurrido un tiempo suficiente.

Referencias

[1] Hambling, B., Morgan, P., Samaroo, A., Thompson, G. y Williams, P., *Software Testing. An ISEB Foundation*. The British Computer Society, 2007.

[2] Koomen, T. y Pol, M., *Test Process Improvement: A practical step-by-step guide to structured testing*, Addison-Wesley, 1999.

[3] Swinkels, R., *Technical Report 12-4-1-FP. A Comparison of TMM and Other Test Process Improvement Models*, Frits Philips Institute, 2000.

[4] Kulkarni, S., "Test Process Maturity Models – Yesterday, Today and Tomorrow". *Proceedings of the 6th Annual International Software Testing Conference, Delhi, India*, marzo, 2006.

[5] Chrissis, M. B., Konrad, M. y Shrum, S., *CMMI: Guidelines for Process Integration and Product Improvement*. Addison-Wesley, 2003.

[6] El Emam, K., Drouin, J.N. y Melo, W., *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*. Wiley-IEEE Computer Society Press, 1997.

[7] Ericson, T., Subotic, A. y Ursing, S, "TIM-a test improvement model. Software Testing", *Verification and Reliability*, vol. 7, num. 4, pp. 229-246, 1997

[8] Burnstein, I., Suwannasart, T. y Carlson, C. "Developing a Testing Maturity Model". *Crosstalk: The Journal of Defense Software Engineering*. Part I, vol. 9, num. 8, pp. 21-24, Part II, vol. 9, num. 9, pp. 19-26, 1996.

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

[10] Brodman, J. G. y Johnson, D. L.., "What Small Businesses and Small Organisations say about CMM?", *Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italia, mayo 1994*, pp. 331 – 340, 1994.

[11] Lied, H.J. y Stlhane, T., "Experience from process improvement in a SME", *Proceedings of the EuroSPI'99 Conference Pori, Finland, 25-27 octubre*, 1999.

[12] McAdam, R., "Quality models in an SME context: A critical perspective using a grounded approach". *International Journal of Quality & Reliability Management*, vol.17, num. 3, pp. 305-323, 2000.

[13] Brown, A., Wiele, T.V.D. y Loughton, K., "Smaller enterprises' experiences with ISO 9000", *International Journal of Quality & Reliability Management*, vol. 15, num. 3, pp. 273-285, 1998.

[14] Rout, T., Tuffley, A, Cahill, B y Hodgen, B., "The Rapid Assessment of Software Process Capability", *Proceedings of SPICE 2000, Limerick*, pp. 47-55, 2000.

[15] Alexandre, S., Renault, A. y Habra, N., "OWPL Software Process Improvement for VSE, SME and low maturity enterprises", *Conference on Software Engineering and Advanced Applications, 2006. SEAA '06. 32nd EUROMICRO*, pp. 328-335, 2006.

[16] Karlström, D., Runeson, P. y Norden, S., "A Minimal Test Practice Framework for Emerging Software Organisations. Software Testing", Verification and Reliability, vol.15, num. 3, pp. 145-166, 2005.

[17] Calvo-Manzano, J. A., Cuevas, G., San Feliu, T., De Amescua, A., García, L. y Pérez, M., "Experiences in the Application of Software Process Improvement in SMES", Software Quality Control, vol. 10, num. 3, pp. 261-273, 2002.