Revista Española de
Innovación, Calidad e Ingeniería del Software

Volumen 2, No. 1, abril, 2006

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

Publicado por la Asociación de Técnicos en Informática

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

Contenidos

REICIS

Editorial

REICIS

REICIS comienza en este número la incorporación de contribuciones de autores que investigan en el campo de la ingeniería y la calidad del software a partir de un proceso de revisión avalado por su comité editorial con evaluaciones rigurosas en lo científico a la vez que centradas en la vertiente práctica de la industria y la realidad empresarial.

REICIS irá extendiendo posteriormente sus fuentes de contribuciones mediante la selección revisión de trabajos seleccionados a partir de los enviados a distintos eventos de interés científico y técnico tanto de ámbito nacional e internacional. A través de diversos acuerdos con los comités organizadores de dichos eventos

En cualquier caso, REICIS invita desde estas líneas a todos los profesionales relacionados con el mundo de la Innovación, Calidad e Ingeniería del Software a que utilicen REICIS como el medio para dar a conocer sus trabajos e investigaciones teniendo las máximas garantías de la profesionalidad con que serán tratados sus trabajos. Podrán encontrar todas las instrucciones necesarias para el envío de sus contribuciones en la página web de la revista: www.ati.es/reicis.

Así mismo REICIS anima a los responsables de eventos técnicos en el ámbito de esta revista a contactar con los editores para posibles acuerdos de publicación de trabajos.

Luis Fernández Sanz
Juan J. Cuadrado-Gallego
Editores

Presentación

REICIS

REICIS reúne en este número dos contribuciones remitidas por grupos investigadores sobre ingeniería y calidad del software existentes en España. Ambos se integran como resultados en proyectos de I+D financiados por diversas entidades públicas.

En el primer artículo, Francisco J. Pino, Félix García y Mario Piattini plantean el reto de adaptar las propuestas de mejora de procesos a las pequeñas y medianas empresas. A través de la técnica de revisión sistemática, los autores, vinculados a los grupos de investigación en ingeniería de software de la Universidad de Castilla-La Mancha, efectúan una revisión profunda de las propuestas publicadas en fuentes de prestigio en el ámbito de la ingeniería del software. Se obtienen interesantes conclusiones sobre los esfuerzos que se realizan en las PYMES aunque no den lugar a certificaciones, sobre los modelos que tienen mayor influencia y sobre el interés mayoritario en abordar la adaptación de los métodos generales a la realidad de estas empresas.

En el segundo artículo se aborda un aspecto de gran interés para una de las principales actividades de aseguramiento de calidad: las pruebas del software. En la contribución remitida por los investigadores del Departamento de Informática de la Universidad de Oviedo, se presenta un método para generar casos de prueba que cumplan con ciertos criterios de cobertura mediante el uso de la técnica de búsqueda dispersa. Los resultados presentados por Blanco, Díaz y Tuya sugieren un mejor comportamiento de esta línea que otros métodos similares.

Luis Fernández Sanz

Revista Española de Innovación, Calidad e Ingeniería del Software, Vol.2, No. 1, 2006

Revisión sistemática de mejora de procesos software en micro, pequeñas y medianas empresas

Francisco J. Pino
Grupo IDIS, Facultad de Ing. Electrónica y Telecomunicaciones, Universidad del Cauca
fjpino@unicauca.edu.co

Félix García, Mario Piattini
Grupo Alarcos, Escuela Superior de Informática, Universidad Castilla-La Mancha
{Felix.García, Mario.Piattini}@uclm.es

Abstract

Small and medium enterprises are a very important piece in the gear of the world economy. The software industry in most of countries is composed of an industrial scheme mainly of small and medium software organizations – SMEs. To strengthen this type of organizations, efficient Software Engineering practices adapted to their size and business type are necessary. The Software Engineering community in the latest decade has expressed a special interest in software process improvement (SPI) with the purpose of improving the quality and productivity of the software. Nevertheless, there is a widespread tendency to stand out that the success of SPI is only possible for large companies. In this article a systematic review of the literature on the SPI efforts carried out in SMEs is presented. The objective is to know what has been carried out and achieved about software process improvement in this type of companies.

Resumen

Las micro, pequeñas y medianas empresas -Pymes- son una pieza muy importante en el engranaje de la economía mundial. La industria del software en la mayoría de los países está formada por tejido industrial compuesto en gran parte por Pymes desarrolladoras de software. Para fortalecer este tipo de organizaciones se necesitan prácticas eficientes de Ingeniería del Software adaptadas a su tamaño y tipo de negocio. La comunidad vinculada a esta disciplina ha expresado en la última década especial interés en la mejora de procesos software con el fin de aumentar la calidad y productividad del software. Sin embargo, hay una tendencia generalizada a resaltar que el éxito de los programas de mejora de procesos software sólo es posible para empresas grandes. En este artículo se presenta una revisión sistemática de la literatura sobre los esfuerzos llevados a cabo en Pymes desarrolladoras de software relacionados con la mejora de sus procesos. El objetivo es conocer lo que se ha realizado y logrado en este tipo de empresas respecto a la mejora de procesos software.

Palabras clave: Mejora de procesos software, SPI, Pymes, Revisión sistemática.

1. Introducción

Las micro, pequeñas y medianas empresas -Pymes- son una pieza muy importante en el engranaje de la economía mundial. En las últimas dos décadas la industria del software ha emergido, crecido y fortalecido a tal punto que representa actualmente una actividad económica de suma importancia para todos los países del mundo. La industria del software en la mayoría de los países esta formada por tejido industrial compuesto en gran parte por Pymes desarrolladoras de software -Pymes_DS- que favorecen al crecimiento de las economías nacionales. Según [13] la mayoría de empresas desarrolladoras de software son pequeñas (tienen menos de 50 empleados) y desarrollan productos significativos que, para su construcción, necesitan prácticas eficientes de Ingeniería del Software adaptadas a su tamaño y tipo de negocio.

A partir de principios de los años noventa la comunidad de Ingeniería del Software (industria e investigadores) ha expresado especial interés en la mejora de procesos software, conocida internacionalmente como SPI *(Software Process Improvement)*. Esto se evidencia por el creciente número de artículos que tratan el tema según el análisis de la tendencia de las publicaciones de mejora de proceso presentado en [14], así como por la aparición de un gran número de iniciativas internacionales relacionadas con SPI, entre las que se encuentran CMM[19], CMMI[5], ISO/IEC 15504 [7][8], SPICE (ISO/IEC TR 15504:1998) [2] e ISO/IEC 12207:2004[6]. Además la norma ISO 9001:2000 [4] está siendo utilizada para ser aplicada en este campo.

En la academia y en la industria hay una tendencia generalizada a resaltar que los programas SPI exitosos sólo son posibles para empresas grandes que cuentan con los recursos suficientes para embarcarse en este tipo de prácticas. Tal percepción se basa en que los programas SPI son prohibitivos para las Pymes_DS debido a la estructura organizacional de dichas empresas, al costo que los programas de mejora implican y a que los estándares de mejora propuestos internacionalmente por organismos como el Software Engineering Institute (SEI) e International Organization for Standarization (ISO) no han sido creados para éste tipo de empresas sino para empresas grandes [15][23]. “Casi todos los autores están de acuerdo en que las características especiales de las pequeñas empresas

hacen que los programas de mejora de procesos deban aplicarse de un modo particular y visiblemente diferente a como se hace en las grandes organizaciones y que esto no es tan sencillo como el hecho de considerar dichos programas de mejora versiones a escala de las grandes compañías" [17][21].

En este artículo se presenta una revisión sistemática de la literatura acerca de los esfuerzos SPI llevados a cabo en Pymes_DS, con el objetivo de conocer qué se ha realizado y logrado sobre mejora de procesos software en este tipo de empresas. Además de la presente introducción el artículo muestra en la sección 2 una visión general del método para realizar la revisión sistemática. En la sección 3 se introduce la revisión sistemática de SPI en Pymes_DS. La sección 4 presenta los resultados y una discusión de ellos. La sección 5 muestra las conclusiones y trabajos futuros.

2. Revisión sistemática en Ingeniería del Software

Una revisión sistemática de la literatura permite identificar, evaluar, interpretar y sintetizar todas las investigaciones existentes y relevantes en un tema de interés particular. Este tipo de revisiones se ejecutan de forma rigurosa e imparcial para que tengan un alto valor científico. La principal motivación para emprender una revisión sistemática es incrementar la posibilidad de detectar más resultados reales en el tema de interés que los que pueden ser detectados con revisiones de menor dimensión.

En [16] Barbara Kitchenham presenta un método para la realización de revisiones sistemáticas en el contexto de Ingeniería del Software, antes de dicha propuesta no se disponía de ninguna guía o método eficiente para realizar estudios exhaustivos en tal contexto. Para la realización de una revisión sistemática se involucran diferentes actividades independientes, para lo cual el método propone tres fases fundamentales. La primera fase es la planificación de la revisión que consta de las actividades de (i) identificación de la necesidad de la revisión y (ii) desarrollo de un protocolo de revisión. La segunda fase es el desarrollo de la revisión que consta de las siguientes actividades: (i) identificación de la investigación, (ii) selección de los estudios primarios, (iii) evaluación de la calidad del estudio, (iv) extracción y monitoreo de datos, y (v) síntesis de datos. La tercera fase es la publicación de los resultados de la revisión.

Para facilitar la planificación y ejecución de la revisión sistemática en [10] se ha desarrollado una plantilla del protocolo para la revisión. El objetivo es que sirva como guía a los investigadores en Ingeniería del Software al conducir revisiones sistemáticas.

3. Revisión sistemática de SPI en Pymes_DS

Para llevar a cabo la revisión sistemática de SPI en Pymes_DS se siguió la plantilla del protocolo presentado en [10]. El protocolo para la revisión sistemática consta de cinco partes generales: formulación de la pregunta, selección de las fuentes, selección de los estudios, extracción de información y resumen de los resultados. En esta sección se aborda las cuatro primeras partes de manera general.

3.1 Formulación de la pregunta

La pregunta de investigación es ¿Cuales son las iniciativas que han sido llevadas a cabo para SPI centradas en Pymes_DS y que al mismo tiempo presentan un caso de estudio real? La lista de los términos usados para resolver la pregunta de investigación fueron: *software*, *process*, *improvement*, *SPI*, *small*, *enterprises*, *organizations*, *companies*, *team*, *firms*, *SME*, *settings*, *CMM*, *CMMI*, *15504*, *SPICE*, *12207* y *9001*.

Los resultados esperados al finalizar la revisión sistemática fueron, entre otros, la identificación de cuáles son: los procesos mejorados, los factores claves para la mejora, las estrategias de mejora llevadas a cabo y los estándares para la mejora de procesos más utilizados. La población a observar fueron las publicaciones relacionadas con SPI en Pymes_DS, existentes en las fuentes seleccionadas para la revisión.

3.2 Selección de las fuentes

A partir de la combinación de la lista anterior de términos con los conectores lógicos “AND” y “OR” se obtuvieron dos cadenas generales básicas de búsqueda (ver tabla 1).

	Cadenas generales básicas de búsqueda
1	"software process improvement" AND (small AND(enterprises OR organizations OR companies OR team OR firms OR settings))
2	(small AND (enterprises OR organizations OR companies OR team OR firms OR settings)) AND (CMM OR CMMI OR 15504 OR SPICE OR 9001 OR 12207)

Tabla 1. Cadenas generales básicas de búsqueda

En el momento de ejecutar las búsquedas dichas cadenas generales tuvieron que ser adaptadas a cada una de los motores de búsqueda de las fuentes escogidas.

La lista de fuentes iniciales a partir de la cual se ejecutó la revisión sistemática fueron: Science@Direct en el tema de Computer Science, Wiley InterScience en el tema de Computer Science, IEEE Digital Library, ACM Digital Library y como literatura gris se revisó el informe técnico "Proceedings of the First International Research Workshop for Process Improvement in Small Settings" del SEI.

3.3 Selección de los estudios

El criterio de la ejecución de la revisión sistemática fue basado en el modelo iterativo e incremental. Iterativo porque la ejecución (búsqueda, extracción de información y visualización de resultados) de la revisión sistemática se hace primero completamente en una fuente de búsqueda, y así sucesivamente sobre las demás. Incremental porque el documento (que es el producto) de la revisión sistemática va creciendo y evolucionando en cada iteración hasta convertirse en el definitivo. De esta forma se obtuvieron todas las ventajas que ofrecen este tipo de modelos de desarrollo. El procedimiento para la obtención de los estudios primarios se describe en el diagrama de flujo de la figura 1.

El criterio de inclusión de los estudios fue basado en el análisis del título, abstract y palabras claves de los artículos obtenidos en la búsqueda para determinar si la propuesta presentada estaba enfocada sobre SPI en Pymes_DS y además había sido aplicada a través de un caso de estudio real. También se analizó cómo se trataban las palabras clave improvement y small en el contenido total de cada artículo para decidir si tenía que ser seleccionado en el contexto de la revisión sistemática como estudio relevante (candidato potencial a convertirse en estudio primario).

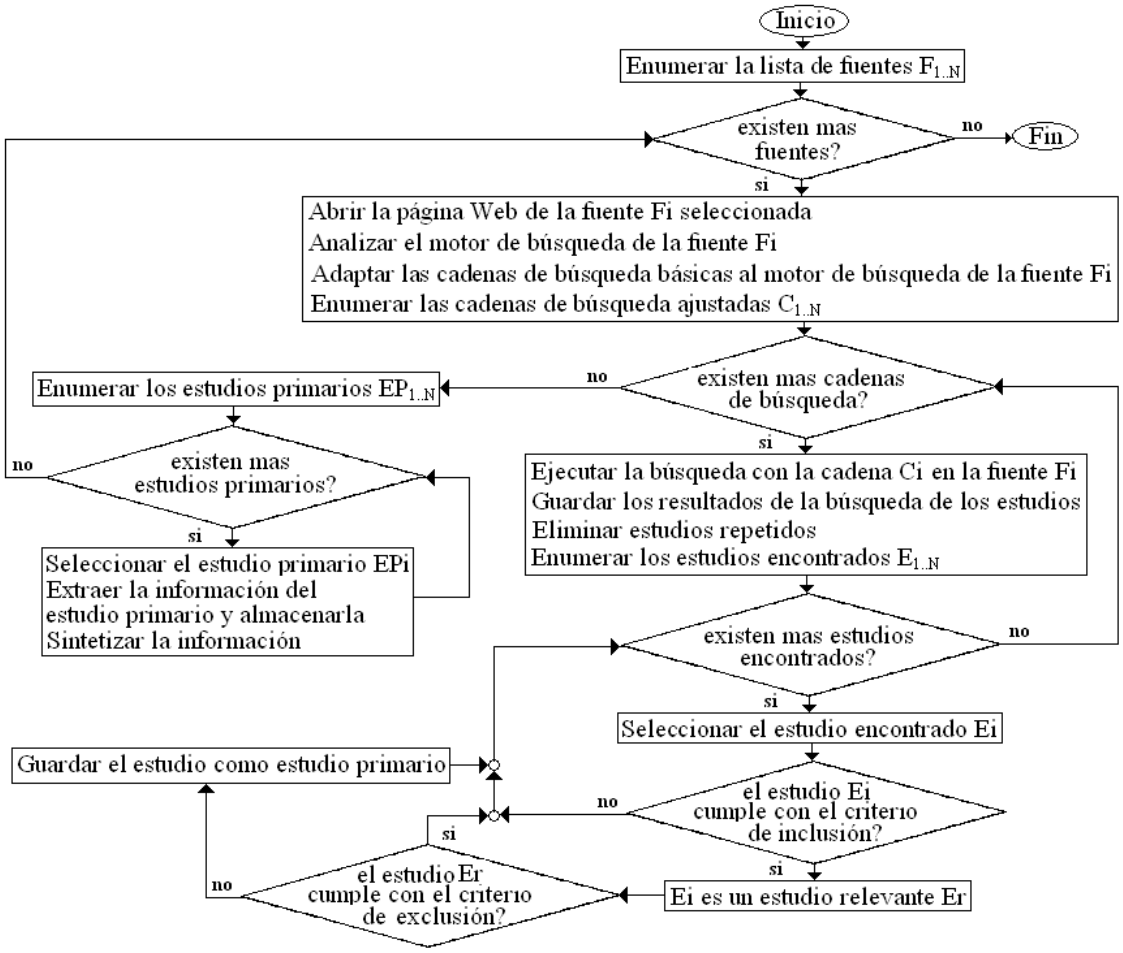

Figura 1. Procedimiento para obtener los estudios primarios y sintetizar su información

Del análisis realizado en la primera iteración de los estudios relevantes se identificó que éstos hacían diferentes niveles de tratamiento de SPI y Pymes_DS:

- Un primer nivel donde el estudio consiste en la aplicación de un esfuerzo de mejora de procesos centrado en pequeñas y medianas organizaciones.
- Un segundo nivel donde el estudio trata iniciativas de SPI para Pymes_DS, pero que no han sido aplicadas en dichas empresas.
- Un tercer nivel donde el estudio reporta iniciativas SPI donde se involucran algunas Pymes_DS pero la mejora en estas no es su objetivo central.

Para determinar qué artículos relevantes eran suficientemente importantes en el contexto de la revisión sistemática para ser considerados como estudios primarios se

definió como criterio de exclusión de los estudios a todos los estudios que trataban SPI y Pymes_DS en los niveles dos y tres.

Después de aplicar el procedimiento para la obtención de estudios primarios se encontraron 847 estudios, 437 estudios no repetidos y de ellos se obtuvieron 45 estudios primarios. La lista completa de los estudios se puede encontrar en [20].

3.4 Extracción de información

Como se puede observar de la figura 1, una vez escogidos los estudios primarios se realizó la extracción de la información relevante para la revisión sistemática. El criterio de inclusión de información a partir de los estudios primarios consistió en obtener información sobre la estrategia de mejora utilizada, los procesos mejorados, los factores claves para el éxito de la mejora y los estándares utilizados para la mejora de procesos, además se registran las ideas más importantes del estudio. La información de las publicaciones primarias se almacenó en una tabla cuyo formato de extracción de datos se estructuró como se muestra a continuación.

Nombre	Practical Software Process Improvement-The IMPACT Project.
Revista	13th Australian Software Engineering Conference (ASWEC'01).
Fecha	Agosto de 2001.
Autores	Louise Scott, Ross Jeffery, Lucila Carvalho, John D'Ambra, Philip Rutherford.
Resultados objetivos del estudio	
Metodología	Caso de estudio
Resultados	• Presenta un framework para mejora de procesos software apropiado para las pequeñas y medianas compañías desarrolladoras de software. • El framework permite que la compañía utilice cualquier tecnología SPI, que sea relevante e importante para sus objetivos de mejora.
País	Australia
No. Pymes	1
No. Empleados	20
Modelos/Estándares	IMPACT, ISO 12207
Propuesta de mejora	Utilizar un framework para guiar el programa SPI
Claves de éxito	Guiar el programa de mejora. Libertad en escoger la tecnología de mejora.
Procesos mejorados	Documentación de proyectos, planeación de procesos y estimación de procesos, elicitación de requisitos.
Resultados subjetivos	
• Para las SMEs los requisitos más urgentes para los paradigmas de mejora son que ellos no sólo sean eficaces, sino que ellos generen resultados tangibles rápidamente, puedan llevarse a cabo incrementalmente y puedan utilizar la mayoría de tecnologías de mejora de proceso existentes. • La cultura de mejora debe seguir la idea de la cultura de desarrollo a pequeña escala. Es decir: plazos cortos, proyectos dinámicos y presupuestos apretados. • El framework es cíclico – y aboga por "entienda - mejore - aplique - mida" el acercamiento que se puede aplicar es incremental a través de muchos proyectos.	

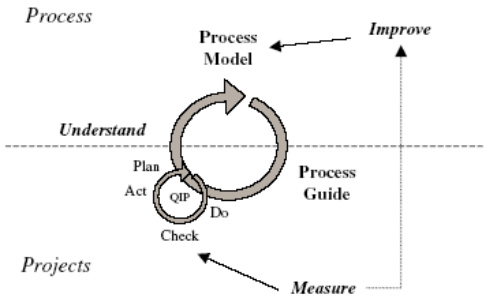

Figure 1: A framework for software process improvement.

- Por implementar muchas iteraciones pequeñas de los ciclos de mejora los resultados son logrados rápidamente y medibles en proyectos en los cuales la mejora ha sido aplicada.
- Los efectos tangibles de mejora han sido: recolección y documentación de requerimientos, planeación y estimación de procesos mas estructurado, mayor conciencia de la importancia del proceso en la compañía.

Tabla 2. Ejemplo del formato de extracción de información

4. Resultados y discusión

A partir de la información extraída de los estudios se realizó un análisis estadístico para mostrar descubrimientos relevantes de la revisión sistemática. A continuación se muestran los resultados desde diferentes puntos de vista.

4.1 Tendencia de las publicaciones

En primer lugar cabe destacar que hay un especial interés en la comunidad de Ingeniería del Software en abordar SPI en Pymes_DS, evidenciado por el creciente número de estudios que tratan el tema según el análisis de la tendencia de las publicaciones realizado (ver figura 2).

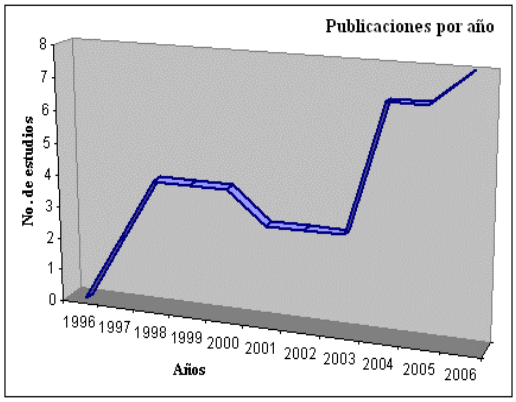

Figura 2. Tendencia de las publicaciones SPI en Pymes_DS

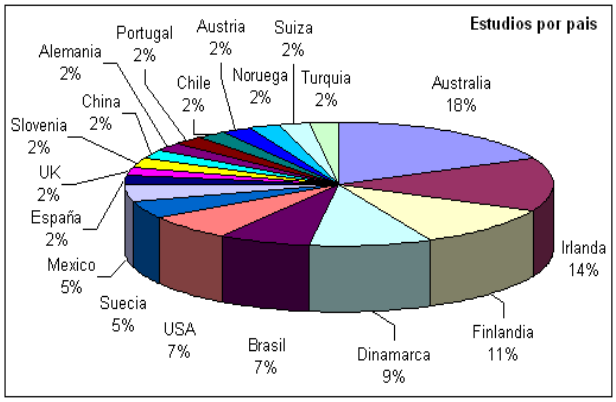

Figura 3. Distribución de los reportes de mejora por país

Como se puede apreciar en la Figura 3, en orden descendente los seis primeros países que reportan más esfuerzos de mejora en Pymes_DS son Australia, Irlanda, Finlandia, Dinamarca, Brasil y USA. Es importante resaltar que el fortalecimiento de la industria de software involucra el compromiso real de diferentes actores de la sociedad como son el Estado, la academia y las empresas. La Unión Europea ha impulsado iniciativas como ESSI (European Software and System Initiative) que han promovido diferentes proyectos para fortalecer SPI en Pymes_DS, como por ejemplo SPIRE (Software Process Improvement in Regions of Europe)[1], TOPS (Toward Organised Software Processes in SMEs) [3], entre otros. En Brasil el gobierno subsidió la implementación del programa PBQP-Software (Productivity and Quality Software Program) [9] y se ha desarrollado el proyecto ”mps Br” (melhoria do processo de software brasileiro) [26] que persiguen el mismo objetivo. Muchos de los estudios reportados están enmarcados dentro de estrategias impulsadas por organismos gubernamentales, así mismo es fundamental el compromiso de la empresa y su confianza en que emprender un proyecto SPI les aportara ventajas competitivas.

4.2 Empresas involucradas

El promedio de Pymes_DS involucradas por estudio es de 2,97. Es decir en cada uno de los estudios se utiliza en promedio tres empresas. Sin embargo es importante tener en cuenta que el 53 % de los estudios involucraron una sola empresa.

Analizando la evolución histórica de las actividades de mejora de los procesos de software, en un principio éstas eran sólo posibles para las grandes empresas. Luego se introdujeron en las empresas de tamaño mediano y ahora los esfuerzos SPI están siendo aplicados a las micro y pequeñas empresas (ver figura 4). Prueba de ello es que el 47% de las empresas involucradas en los esfuerzos de mejora eran pequeñas (entre 10 y 49 empleados) y el 33% eran micro (entre 1 y 9 empleados), lo que equivale al 80% de todas las empresas reportadas en las que se llevó a cabo algún esfuerzo de SPI. Sólo se reporta un 20 % en medianas empresas (entre 50 y 249 empleados). Sin embargo es importante resaltar que en la mayoría de los casos los programas de mejora en micro y pequeñas empresas de software no condujeron a corto plazo a la certificación del ISO y menos aún a la certificación del SEI.

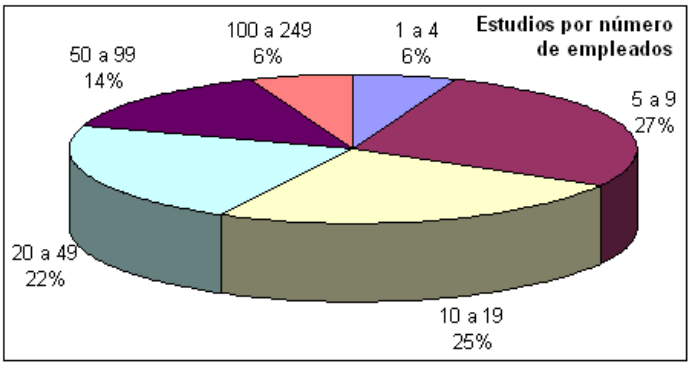

Figura 4. Número de empleados de las empresas donde se realizó un esfuerzo de mejora

4.3 Estándares utilizados

Muchos autores coinciden en que los estándares del ISO y modelos del SEI (considerados como estándares de facto) difícilmente pueden ser aplicados a empresas pequeñas debido a que un proyecto de mejora supone gran inversión en dinero, tiempo y recursos [15][23]. Sin embargo las Pymes_DS adaptan y utilizan estos estándares (de hecho y de facto) para emprender sus esfuerzos de mejora. En la figura 5 se muestran todos los estándares utilizados en los esfuerzos de mejora.

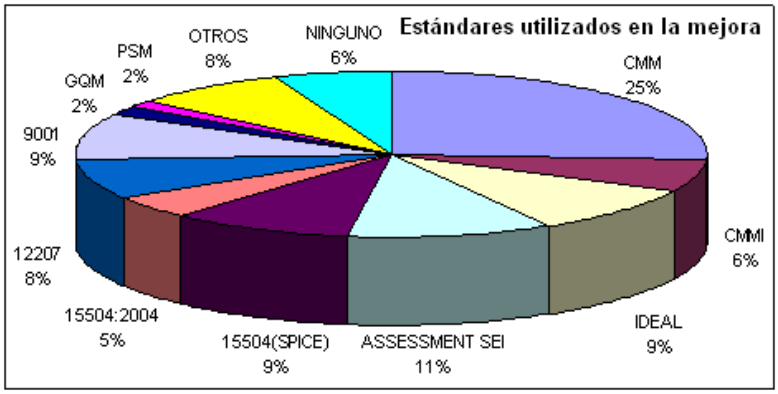

Figura 5. Estándares utilizados para la mejora

4.4 Modelos que guían la mejora, métodos de evaluación y modelos de procesos

Es importante resaltar que en un programa de mejora se involucran diferentes tipos de modelos/métodos, entre los que se encuentran el modelo que conduce la mejora, el método

de evaluación de procesos y el modelo de procesos a seguir. El modelo de mejora describe la infraestructura, actividades, ciclo de vida y consideraciones prácticas para la evolución de los procesos. El método de evaluación de procesos especifica la ejecución de la evaluación para producir un resultado cuantitativo que caracterice la capacidad del proceso o la madurez de la organización. El modelo de procesos de referencia describe cuáles son reconocidas como las mejores prácticas que una organización debe implementar para el desarrollo de software.

La adaptación del modelo IDEAL [18] es la forma más utilizada para conducir la mejora, el estudio presentado en [12] es un buen ejemplo de esto. También se utilizan modelos como IMPACT [24] y MESOPYME [11], entre otros. Por ejemplo MESOPYME se centra en reducir el tiempo y el esfuerzo en la implementación de SPI basado en el concepto de paquetes de acción. Este tipo de modelos se utilizaron en 9 estudios, es decir en sólo el 20 % de los estudios, lo que es un porcentaje bajo, ya que un modelo de mejora constituye la guía necesaria [0]para articular todas las actividades relacionadas con la mejora y por supuesto todos los demás modelos involucrados.

El método de valoración más utilizado en las Pymes_DS es el estándar ISO/IEC 15504, considerando sus versiones de 1998 y de 2004. Comienza a tener una gran aplicación el estándar 15504:2004 en las Pymes_DS, debido principalmente a que es flexible y fácil de adecuar a sus necesidades [25]. En algunos estudios se presenta la utilización de modelos de valoración desarrollados particularmente para pequeñas empresas y basados en el estándar ISO 15504, como por ejemplo MARES [25] y RAPID [22]. Por su parte del SEI se siguen valoraciones como la propuesta por CBA-IPI o valoraciones menos intensivas como las de clase C o *micro-appraisal*.

Con respecto al modelo de procesos se observa que el más utilizado es el modelo CMM del SEI. En el ámbito del CMM las Pymes_DS tienen como meta, cuando comienzan un programa de mejora, lograr el nivel de madurez 2. Es importante resaltar que de las 122 empresas involucradas en los estudios primarios sólo dos medianas empresas de 70 y 150 empleados lograron evaluación formal como CMM-SW Nivel 2. Para las pequeñas y micro empresas los estándares del SEI no condujeron a una certificación. Además el 30% de los esfuerzos de mejora que utilizaron CMM también utilizaron ISO 9001 (en muchos países es una norma nacional) la cual ayuda a proporcionar un mejor

soporte a la iniciativa CMM para mejorar las áreas de la organización que no están directamente relacionadas con el desarrollo del software.

4.5 Medición de procesos

De los estudios analizados 20 han utilizado modelos del SEI y 18 han utilizado estándares del ISO, de éstos últimos dos utilizaron procesos de medición formal como Goal Question Metric (GQM) o Practical Software Measurement (PSM). Generalmente las mejoras introducidas por esfuerzos SPI en Pymes_DS se miden a través de procesos informales y subjetivos basados en la percepción de los empleados y no a través de procesos formales de medición, como por ejemplo GQM o PSM, con los cuales la evaluación sería más objetiva. Esto confirma que aún son pocos los esfuerzos de las Pymes_DS para establecer procesos formales de medición de software.

4.6 Propuestas de mejora

Las propuestas de mejora sugeridas para guiar el esfuerzo de mejora son variadas. En la siguiente tabla se muestran las propuestas de mejora encontradas en este estudio.

	Propuestas de mejora
18%	Establecimiento de procesos software. Usar una guía electrónica de proceso – ERP y repositorio de experiencias - ER. Adaptar y utilizar practicas de RUP, XP, SCRUM, entre otras. Autovaloración de procesos por los empleados (herramienta WEB).
9%	Priorizar los esfuerzos de SPI (a través del método DAIIPS, Software Process Matrix, Express Process Appraisal ó framework para tomar decisiones de negocio y producto)
4%	Evaluación de un programa SPI (con poca rigurosa, ó definir y usar un programa de métricas)
13%	Guiar los esfuerzos de SPI (método Pr2imer, Framework IMPACT, método/modelo MESOPYME, usando patrones de mejora, siguiendo un proceso de valoración, redes neuronales)
45%	Adaptación y utilización de estándares SPI (PSP, TSP, CMM, CMMI, IDEAL, ISO 15504:2004, SPICE, ISO 9001)
11%	Definición y utilización de framework para pruebas. Conducción de un experimento de SPI. Usar gestión de conocimiento para SPI. Adquirir infraestructura técnica lista para usar. Mejorar la relación y cooperación con el cliente.

Tabla 3. Propuestas de mejora

De la tabla anterior se puede observar que la mayor concentración de propuestas de mejora (71%) radica en guiar un proyecto de mejora, priorizar la implementación de las mejoras, utilizar modelos de mejora existentes ajustándolos a las necesidades y evaluar las

mejoras introducidas por el programa SPI. Además hay otras propuestas de mejora (18%) centradas en la definición, evaluación y soporte de los procesos software.

Los enfoques de mejora pueden darse a nivel organizacional y/o a nivel técnico. A nivel organizacional son más económicos pero las mejoras no se ven a corto plazo. A nivel técnico son más costosas pero las mejoras se ven a corto plazo. Es importante resaltar que hay estudios (11%) que para soportar acciones de mejora de proceso proponen suministrar herramientas técnicas e instalaciones que apoyan actividades relacionadas con procesos, a lo cual se le denomina infraestructura técnica, como por ejemplo plantillas de documentos, modelos de proceso, técnicas y herramientas.

4.7 Procesos mejorados

De los estudios primarios se buscó la información que reportaba de manera explícita los procesos mejorados. Dicha información se almacenó en el formato de extracción de datos descrito en la sección 3.4. Luego tal información relacionada con la mejora y que estaba expresada en lenguaje natural se analizó y adaptó para poderla encuadrar en un modelo de procesos. Esto permitió tener una visión sobre hacia dónde se han enfocado lo esfuerzos de mejora en las micro, pequeñas y medianas empresas. El estándar ISO/IEC 12207:2004 [6] se utilizó como modelo de procesos para expresar las mejoras. En la figura 6 se muestran las frecuencias de los procesos mejorados.

Como se puede apreciar en la figura 6, los esfuerzos de mejora apuntan a mejorar procesos como la gestión del proyecto, documentación, gestión de cambio de requisitos, establecimiento de procesos, gestión de la configuración y obtención de requisitos. Hay otros procesos como por ejemplo el de revisión conjunta, usabilidad, adquisición, suministro, entre otros, sobre los cuales no se reporta ningún tipo de mejora.

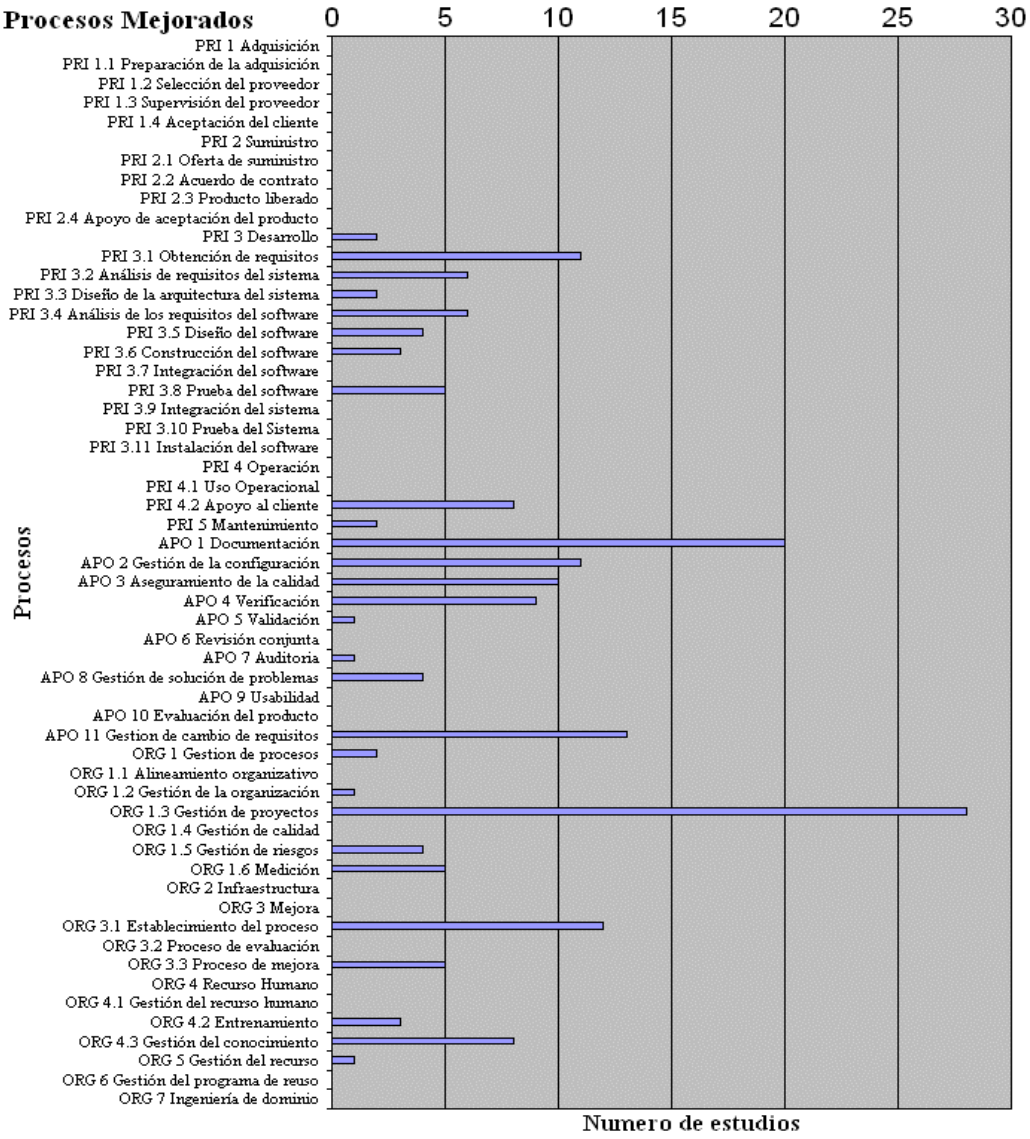

Figura 6. Proceso mejorados en las Pymes_DS con los esfuerzos de mejora.

4.8 Factores de éxito

Algunos factores de éxito de los esfuerzos SPI extraídos de los estudios analizados son:

- Asegurarse que la Pyme es estable para iniciar un programa SPI. Iniciar lo más rápido posible la mejora con una estructura simple del modelo SPI. Guiar el programa de mejora mediante procedimientos concretos, combinando diferentes enfoques, y siguiendo una iniciativa sistemática y coherente. Mejorar los procesos siguiendo una

aproximación incremental que permita una adopción continúa de las prácticas de mejora. Priorizar los aspectos de mejora en un programa SPI.

- Conseguir un rápido retorno a la inversión, extendiendo al máximo los recursos asignados y maximizando las mejoras en el menor tiempo posible. Financiar y compartir con otras empresas los recursos especializados involucrados en la mejora. Gestionar la adquisición de ayuda financiera externa.
- Minimizar la resistencia al cambio mediante la concientización organizacional de que la mejora de procesos, implementándola con base en las necesidades reales, beneficiará a los empleados y a la empresa.
- Involucrar a todas las personas de la empresa en la búsqueda permanente de la calidad, minimizando las interrupciones del programa mediante la capacitación en el área SPI de los empleados.
- Monitorear y supervisar el programa SPI, evaluando frecuentemente su eficiencia. Una adecuada estrategia es la valoración rápida de procesos software.
- Establecer un mecanismo de comunicación eficiente que soporte la comunicación entre los diferentes actores involucrados en la mejora.
- Llevar a cabo actividades de medición, a través del uso sistemático de métricas adaptadas a la organización.
- Comprometer a las directivas de la empresa en el programa SPI.
- Lograr la asesoría de un experto para iniciar el programa SPI.
- Abordar el problema de mejora desde la perspectiva técnica.

5. Conclusiones y trabajo futuro

En este artículo se ha presentado una revisión sistemática sobre los esfuerzos SPI llevados a cabo en Pymes desarrolladoras de software, que permite tener una visión muy completa de la situación actual. La formalidad con la que se lleva a cabo la revisión sistemática permite validar sus resultados, los cuales están soportados y avalados por el protocolo de la revisión. Las revisiones sistemáticas requieren un esfuerzo considerablemente superior que las revisiones de literatura convencionales.

Para el fortalecimiento de la industria del software, soportada en gran parte por la Pymes_DS, se debe comprometer al Estado, a la academia y a las mismas empresas

enfocando sus esfuerzos para que los estándares del ISO y SEI puedan ser aplicados de manera apropiada en las micro, pequeñas y medianas empresas ya que actualmente dichas empresas adaptan y utilizan tales estándares para emprender sus esfuerzos de mejora, aunque por lo general no consiguen una certificación.

Otro dato relevante obtenido es que de los diferentes tipos de estándares disponibles relacionados con las mejoras de procesos los más utilizados por las Pymes_DS son: CMM como modelo de procesos, ISO/IEC 15504 como método de evaluación e IDEAL como modelo para guiar la mejora.

Con relación a las propuestas de mejora, cabe destacar que aunque son diversas, la mayor concentración de ellas radica en guiar el proyecto de mejora utilizando modelos de mejora existentes ajustados a sus necesidades, así como priorizar la implementación de las mejoras introducidas por el programa SPI evaluándolas con frecuencia. En menor frecuencia hay otras propuestas de mejora centradas en la definición, evaluación y soporte de los procesos software, así como emprender acciones de mejora enfocadas a suministrar infraestructura técnica que apoyan actividades relacionadas con los procesos. Las mejoras introducidas por esfuerzos SPI se miden a través de procesos informales y subjetivos basados en la percepción de los empleados.

A partir de los resultados de esta revisión sistemática se abordarán dos importantes líneas de trabajo. En primer lugar, se realizará con el mismo protocolo de revisión la búsqueda en nuevas fuentes de información de forma que la revisión abarcaría un mayor número de trabajos, al no estar muchos de ellos publicados en las fuentes tratadas en este estudio. En segundo lugar, se analizarán también los estudios que tratan iniciativas de SPI para Pymes_DS, pero que no han sido aplicadas a ellas, y que, de acuerdo al alcance del presente trabajo, han sido excluidas del estudio.

Agradecimientos

Este trabajo ha sido realizado en el Proyecto COMPETISOFT (506PI0287) financiado por el Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo – CYTED**.**

Referencias

[1] *Software Process Improvement in Regions of Europe (SPIRE). European Commission ESPRIT/ESSI Programme*. 1993; Available from: http://www.cse.dcu.ie/spire/.

[2] *ISO/IEC TR 15504:1998(E). Information Technology -Software process assessment. Parts 1-9*. 1998, International Organization for Standardization: Geneva.
[3] *Toward Organised Software Processes in SMEs. Esprit Project 27977 - TOPS.* 1999; Available from: http://www.cordis.lu/esprit/src/27977.htm.
[4] *ISO 9001:2000. Quality management systems -Requirements.* 2000, International Organization for Standardization: Geneva.
[5] *CMMI for Systems Engineering/Software Engineering, Version 1.1*. 2002, Software Engineering Institute: Pittsburgh.
[6] *ISO/IEC 12207:2002/FDAM 2. Information technology - Software life cycle processes*. 2004, International Organization for Standardization: Geneva.
[7] *ISO/IEC 15504-2:2003/Cor.1:2004(E). Information technology - Process assessment - Part 2: Performing an assessment*. 2004, International Organization for Standardization: Geneva.
[8] *ISO/IEC 15504-5:2006(E). Information technology - Process assessment - Part 5: An exemplar Process Assessment Model*. 2006, International Organization for Standardization: Geneva.
[9] Bedini, A., et al. *Quality Software Map of South America*. in *Proceedings of the First International Research Workshop for Process Improvement in Small Settings*. 2005. Pittsburgh. p. 216-227.
[10] Biolchini, J., et al., *Systematic Review in Software Engineering*. 2005, Systems Engineering and Computer Science Department, UFRJ: Rio de Janeiro, Brazil.
[11] Calvo-Manzano, J.A., et al., *Experiences in the Application of Software Process Improvement in SMES.* Software Quality Journal, 2002. **10**(3): p. 261-273.
[12] Casey, V. and I. Richardson, *A practical application of the IDEAL model.* Software Process: Improvement and Practice, 2004. **9**(3): p. 123-132.
[13] Fayad, M.E., M. Laitinen, and R.P. Ward, *Software Engineering in the Small.* Communications of the ACM, 2000. **43**(3): p. 115-118.
[14] Hall, T., A. Rainer, and N. Baddoo, *Implementing Software Process Improvement: An Empirical Study.* Software Process: Improvement and Practice, 2002. **7**(1): p. 3-15.
[15] Hareton, L. and Y. Terence, *A process framework for small projects.* Software Process: Improvement and Practice, 2001. **6**(2): p. 67-83.
[16] Kitchenham, B., *Procedures for performing systematic reviews (Joint Technical Report)*. 2004, Software Engineering Group, Department of Computer Science, Keele University and Empirical Software Engineering National ICT Australia Ltd.
[17] Mas, A. and E. Amengual, *La mejora de los procesos de software en las pequeñas y medianas empresas (pyme). Un nuevo modelo y su aplicación en un caso real.* Revista Española de Innovación Calidad e Ingenieria del Software (REICIS), 2005. **1**(2): p. 7-29.
[18] McFeeley, R., *IDEAL: A Users Guide for Software Process Improvement, Handbook CMU/SEI-96-HB-001*. 1996, Software Engineering Institute, Carnegie Mellon University: Pittsburgh, USA.
[19] Paulk, M.C., et al., *Capability Maturity Model for Software, Version 1.1 (Technical Report CMU/SEI-93-TR-024)*. 1993, Software Engineering Institute: Pittsburgh.
[20] Pino, F., *Revisión sistemática de mejora de procesos software en micro, pequeñas y medianas empresas. RT Competisoft-001/UCLM*. 2006, Grupo IDIS, Universidad del Cauca y Grupo Alarcos, Universidad Castilla - La Mancha: Ciudad Real, España.

[21] Richardson, I., *Software process matrix: a small company SPI model.* Software Process: Improvement and Practice, 2001. **6**(3): p. 157-165.
[22] Rout, T., et al. *The RAPID Assessment of Software Process Capability*. in *First International Conference on Software Process Improvement and Capability Determination.* 2000. p. 47-56.
[23] Saiedian, H. and N. Carr *Characterizing a software process maturity model for small organizations.* ACM SIGICE Bulletin, 1997. **23**(1): p. 2-11.
[24] Scott, L., et al. *Practical Software Process Improvement -The IMPACT Project.* in *Proceedings of the Australian Software Engineering Conference*. 2001. p. 182-189.
[25] Wangenheim, C.G.v., A. Anacleto, and C.F. Salviano, *Helping Small Companies Assess Software Processes.* IEEE Software., 2006: p. 91-98.
[26] Weber, K., et al., *Brazilian Software Process Reference Model and Assessment Method*, in *Computer and Information Sciences*. 2005, Springer Berlin / Heidelberg. p. 402-411.

Revista Española de Innovación, Calidad e Ingeniería del Software, Vol.2, No. 1, 2006

Generación automática de casos de prueba mediante búsqueda dispersa

Raquel Blanco, Eugenia Díaz, Javier Tuya
Departamento de Informática, Universidad de Oviedo
{rblanco | madiaz | tuya}@uniovi.es

Abstract

The test process is very expensive and the number of test cases needed to test a program is infinite, therefore it is impossible to obtain a fully tested program. For this reason, the techniques for the automatic generation of test cases try to efficiently find a small set of test cases that allow fulfilling an adequacy criterion. A method based on Scatter Search that automatically generates test cases to obtain branch coverage is presented in this article. Besides we show the results we have obtained compared with a method based on Tabu Search.

Resumen

El número de casos de prueba necesarios para probar un programa software es infinito, por lo que es imposible conseguir un programa totalmente probado, siendo además el proceso de prueba muy costoso. Por estos motivos las técnicas para la generación automática de casos de prueba tratan de encontrar de forma eficiente un conjunto pequeño de casos de prueba que permitan cumplir un determinado criterio de suficiencia. En este artículo se presenta un método basado en Búsqueda Dispersa que permite generar automáticamente casos de prueba para obtener cobertura de ramas. Además se muestran los resultados comparativos del método basado en Búsqueda Dispersa y de un método basado en Búsqueda Tabú.

Palabras clave: pruebas de software, automatización de la generación de casos de prueba, cobertura de ramas, técnicas de búsqueda metaheurísticas, búsqueda dispersa.

1 Introducción

La aplicación de algoritmos metaheurísticos para resolver problemas en Ingeniería del Software ha sido propuesto por la red SEMINAL (Software Engineering using Metaheuristic INnovative Algorithms) y se trata ampliamente en [4]. Una de esas aplicaciones es la selección de casos en el proceso de la prueba del software.

La prueba del software es el proceso de ejecutar un programa con el objetivo de encontrar errores [14].El número de casos de prueba necesarios para probar un programa software es infinito, por lo que es imposible conseguir un programa totalmente probado. Además el proceso de prueba es costoso y puede suponer el 50% del coste total del desarrollo software [1]. Por estos motivos las técnicas para la generación automática de casos de prueba tratan de encontrar de forma eficiente un conjunto pequeño de casos de prueba que permitan cumplir un determinado criterio de suficiencia. Entre las técnicas más recientes que son utilizadas para realizar esta automatización se encuentran las técnicas de búsqueda metaheurísticas, como los Algoritmos Genéticos [8][12][13][15][17], el Recocido Simulado [16] o la Búsqueda Tabú [5][6], aunque en la práctica la mayor parte de los trabajos utilizan Algoritmos Genéticos. Otra técnica metaheurística que también puede ser aplicada a la prueba del software es la Búsqueda Dispersa [7][10].

En este artículo se explica un desarrollo específico de la técnica Búsqueda Dispersa para satisfacer el criterio estructural de cobertura de ramas, ampliando lo descrito en [2][3].

2 La técnica de Búsqueda Dispersa

La Búsqueda Dispersa (Scatter Search)[7][10] es un método evolutivo que opera sobre un conjunto de soluciones, llamado Conjunto de Referencia (RefSet).Las soluciones presentes en este conjunto son combinadas con el fin de generar nuevas soluciones que mejoren a las originales. Así, el conjunto de referencia almacena las mejores soluciones que se encuentran durante el proceso de búsqueda, considerando para ello su calidad y la diversidad que aportan al mismo. Esta técnica utiliza estrategias sistemáticas para avanzar en el proceso de búsqueda en vez de aleatorias, siendo ésta una de las principales diferencias con los ampliamente utilizados Algoritmos Genéticos.

El algoritmo Scatter Search comienza generando un conjunto P de soluciones diversas mediante un método de generación de diversidad. Las soluciones presentes en este conjunto pueden ser mejoradas con un método de mejora, el cual es opcional. Posteriormente se construye el conjunto de referencia con las mejores soluciones de P y las más diversas a las ya incluidas. A continuación comienza un proceso cíclico en el cual el algoritmo crea subconjuntos del conjunto de referencia, con un método de generación de subconjuntos, y aplica un método de combinación sobre dichos subconjuntos para obtener las nuevas soluciones. Posteriormente, sobre cada nueva solución aplica un método de mejora y evalúa si debe incorporarse al conjunto de referencia, mediante un método de actualización. El algoritmo se detiene cuando no se generan nuevas soluciones en el proceso de combinación.

3 Descripción del generador de casos de prueba basado en Búsqueda Dispersa

El generador de casos de prueba desarrollado basado en Búsqueda Dispersa se denomina TCSS (Test Coverage Scatter Search) y utiliza el grafo de control de flujo asociado al programa bajo prueba para almacenar información relevante durante el proceso de búsqueda de casos de prueba. Con este grafo es posible determinar qué ramas han sido cubiertas debido a que el programa bajo prueba ha sido instrumentado para determinar el camino seguido por cada caso de prueba ejecutado en él.

El objetivo de TCSS es generar casos de prueba que permitan cubrir todas las ramas de un programa. Este objetivo se puede dividir en subobjetivos, consistiendo cada uno de ellos en encontrar casos de prueba que alcancen un determinado nodo del grafo de control de flujo.

Para alcanzar estos subobjetivos, TCSS trabaja con un conjunto de soluciones (conjunto de referencia) en cada nodo del grafo de control de flujo. Cada uno de estos conjuntos de soluciones se denomina S_k (donde k es el número de nodo) y contiene varios elementos $T_k^c = <\overline{x}_k^c, p_k^c, f_k^c>$, donde $\overline{x}_k^c$ es una solución (un caso de prueba) que alcanza el nodo k, p_k^c es el camino recorrido por la solución y f_k^c es la distancia que indica lo cerca que dicha solución está de pasar por su nodo hermano, es decir, el nodo cuya decisión de entrada es la negación de la decisión del nodo k. Cada una de las soluciones almacenadas en un nodo k está compuesta por los valores de las variables de entrada que hacen ciertas

tanto las decisiones de entrada a los nodos precedentes al nodo k en un determinado camino que permite llegar a él, como la del propio nodo k. La estructura del grafo de control de flujo asociado a un programa, junto con la información que se almacena en cada nodo se puede ver en la Figura 1.

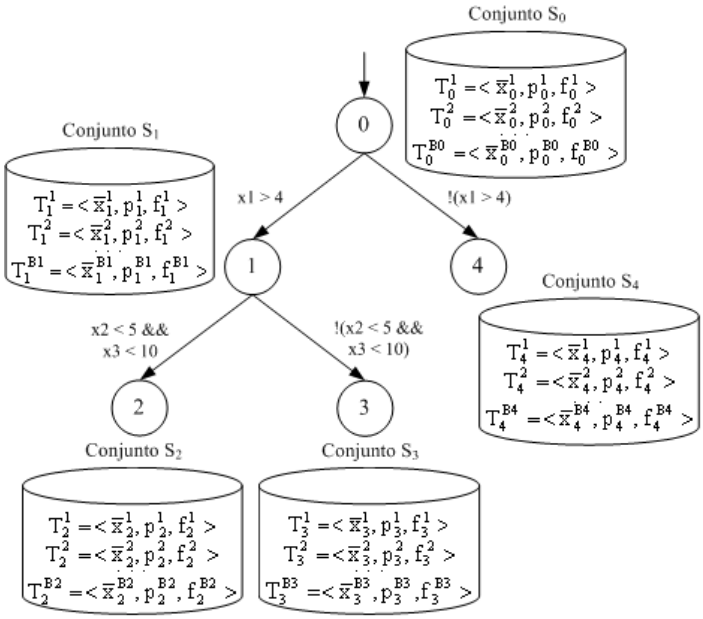

Figura 1. Información almacenada por TCSS en el grafo de control de flujo

El conjunto de soluciones de un nodo k (S_k tiene un tamaño máximo B_k. Este tamaño es distinto para cada nodo k y depende de la complejidad del código situado por debajo de dicho nodo k, que puede ser medida por medio de la complejidad ciclomática [11] del grafo de control de flujo resultante de tomar como raíz al propio nodo k. Este valor de complejidad es multiplicado por un factor fijo para disponer de una cantidad razonable de soluciones en cada conjunto S_k que permita generar otras nuevas.

TCSS tratará de hacer los conjuntos S_k lo más diversos posibles, utilizando una función de diversidad, para generar soluciones que puedan cubrir distintas ramas del programa.

El objetivo de TCSS es obtener la máxima cobertura de ramas, por lo que deben encontrarse soluciones que permitan cubrir todos los nodos del grafo de control de flujo. Como dichas soluciones se almacenan en los nodos, el objetivo de TCSS es, por tanto, que todos los nodos tengan al menos un elemento en su conjunto S_k.

3.1 Cálculo de la distancia asociada a una solución

Para que una solución alcance un determinado nodo, debe cumplirse la decisión de entrada al mismo, por ello, si se desea calcular lo cerca que la solución está de pasar por el nodo hermano se debe utilizar la decisión de entrada a dicho nodo hermano. Por ejemplo, en la Figura 1 la solución $\overline{x}_1^1$ alcanza el nodo 1 y el valor de distancia f_1^1 se calcula utilizando la decisión de entrada al nodo 4 (!(x1>4)). El cálculo de las distancias f_k^c se muestra en la Tabla 1.

Condición	**eval(Condición, $\overline{x}$)**
x=y, x≠y, x<y, x≤y, x>y, x≥y	|x-y|
Decisión	**f_k^c**
C1 AND C2	$\sum \text{eval}(C_j, \overline{x})\ \forall C_j$ False
C1 OR C2	$\text{Min}(\text{eval}(C_j, \overline{x}))\ \forall C_j$
¬C	Aplicar leyes de De Morgan

Tabla 1. Cálculo de las distancias f_k^c

En primer lugar se evalúa cada condición que forma parte de la decisión de entrada al nodo hermano, según los valores de las variables de entrada que constituyen la solución. Posteriormente, se calcula el valor f_k^c de la decisión. Si en la decisión intervienen operadores AND, la distancia f_k^c será el resultado de sumar la evaluación de las condiciones falsas, pues son las que impiden que se alcance al nodo hermano. Si en la decisión intervienen operadores OR, la distancia f_k^c será el valor mínimo de las evaluaciones de las condiciones, ya que todas ellas son falsas y con que se cumpla una sola se alcanzaría el nodo hermano. Si la decisión está negada, simplemente se aplican las leyes de De Morgan.

3.2 Generación de nuevas soluciones

En cada iteración del algoritmo, TCSS selecciona un nodo para generar las nuevas soluciones del proceso de búsqueda por medio de la combinación de las soluciones almacenadas en su conjunto S_k. Los criterios que rigen la selección del nodo con el que trabajar pueden consultarse en [3].

TCSS selecciona un número constante b de elementos $T_k^c = < \overline{x}_k^c, p_k^c, f_k^c >$ del conjunto S_k que previamente no hayan sido utilizados para generar nuevas soluciones, intentado que esas soluciones ($\overline{x}_k^c$) cubran caminos distintos (p_k^c) y tengan menor valor de distancia (f_k^c). Con las soluciones seleccionadas se forman todos los posibles pares ($\overline{x}_k^j$, $\overline{x}_k^h$), con j≠h, y a partir de cada uno de ellos se generan cuatro nuevas soluciones como resultado de aplicar las siguientes combinaciones elemento a elemento: $\overline{x}_k^{ji} \pm \Delta_i$, $\overline{x}_k^{hi} \pm \Delta_i$, donde $\Delta_i = |\overline{x}_k^{ji} - \overline{x}_k^{hi}|/2$ y el índice i recorre todas las variables de entrada. Cada nueva solución es examinada para determinar si los valores de sus variables de entrada se encuentran situados dentro del rango que cada una de ellas puede tomar. De no ser así, se debe aplicar sobre dicha solución el método de mejora, que se limita a modificar los valores de las variables para que no sea considerada inválida.

Con estas combinaciones se pretende generar soluciones que se alejen de las originales y soluciones que se sitúen entre ellas. Además con los criterios de selección de las soluciones se intentan combinar soluciones diversas (recorren distintos caminos) que estén próximas a producir un salto de rama.

Si el nodo seleccionado para generar nuevas soluciones no posee al menos dos soluciones que puedan ser empleadas para realizar las combinaciones se lleva a cabo un proceso de backtracking que puede ser consultado en [3].

El programa bajo prueba es ejecutado con cada nueva solución. Cada una de estas ejecuciones puede causar la actualización de los conjuntos S_k de los nodos alcanzados. Esta actualización se describe en la siguiente subsección.

3.3 Actualización de los conjuntos de soluciones

La actualización de un conjunto de soluciones S_k tiene en cuenta el estado de dicho conjunto. Así si el conjunto S_k no ha excedido su tamaño máximo B_k, la nueva solución es aceptada. En otro caso, la actualización se realiza mediante la función de diversidad. Esta función determina si una nueva solución puede ser añadida a un determinado conjunto S_k y, en ese caso, qué solución debe ser eliminada del mismo.

Cuando se incluye una nueva solución en un conjunto S_k y su tamaño máximo B_k ha sido sobrepasado, se aplica la función de diversidad para determinar la solución que debe abandonarlo, pudiendo ser incluso la solución que provocó el desbordamiento del conjunto.

La función de diversidad se aplica sobre el subconjunto $S_{p^*}=\{T_{p^*}{}^1=<\overline{x}_{p^*}{}^1;p_{p^*};f_{p^*}{}^1>,...,T_{p^*}{}^q=<\overline{x}_{p^*}{}^q;p_{p^*};f_{p^*}{}^q>\} \subseteq S_k$, que representa las soluciones almacenadas en el nodo k que cubren el camino (p_{p^*}) con más ocurrencias en el conjunto S_k. La solución más similar al resto de soluciones que recorren el mismo camino (la menos diversa) abandonará el conjunto S_k. El valor de diversidad de una solución se calcula según la función de diversidad definida como:

$$div(<x^m_{p^*}; p_{p^*}>; S_{p^*}) = \sum_{y=1..q}\left(\sum_{i=1..n}\frac{\left|\overline{x}^{m_i}_{p^*} - \overline{x}^{y_i}_{p^*}\right|}{range_i}\right)$$

donde el índice y recorre las soluciones del conjunto S_{p^*}, el índice i recorre las variables de entrada y $rango_i$ es el rango de valores de la variable de entrada i.

Si existen dos o más soluciones con el mismo valor de diversidad, la solución que será eliminada del conjunto S_k será aquella con mayor valor de distancia, es decir, la menos cercana a producir un salto de rama.

La aplicación de la función de diversidad sobre el subconjunto de soluciones que recorren el camino con más ocurrencias dentro del conjunto S_k tiene como objetivo equilibrar el número de soluciones que cubren diferentes caminos, consiguiendo así mayor diversidad.

4 Resultados

En esta sección, se presentan los resultados obtenidos con un programa que determina la posición de una línea respecto a un rectángulo. Su grafo de control de flujo aparece en la Figura 2.Este programa tiene ocho variables de entrada enteras, cuatro de ellas (xr1, xr2, yr1, yr2) representan las coordenadas de un rectángulo, y las otras cuatro (xl1, xl2, yl1, yl2) representan las coordenadas de la línea. El grafo de control de flujo de este programa tiene nodos difíciles de alcanzar debido a la existencia de igualdades y decisiones compuestas a un alto nivel de anidamiento.

La comparación de TCSS ha sido realizada respecto al generador basado en Búsqueda Tabú TSGen [5][6]. Para ambos generadores se muestran resultados para diferentes rangos (16 bits, 24 bits, 32 bits) y variables de tipo entero. En los experimentos, el criterio de parada utilizado para ambos generadores ha sido alcanzar el 100% de

cobertura de ramas o generar 1.000.000 de casos de prueba. Para cada rango se han llevado a cabo 10 ejecuciones, presentando como resultado el promedio de casos de prueba y tiempo que cada generador necesita para obtener un determinado porcentaje de cobertura de ramas para el programa bajo prueba.

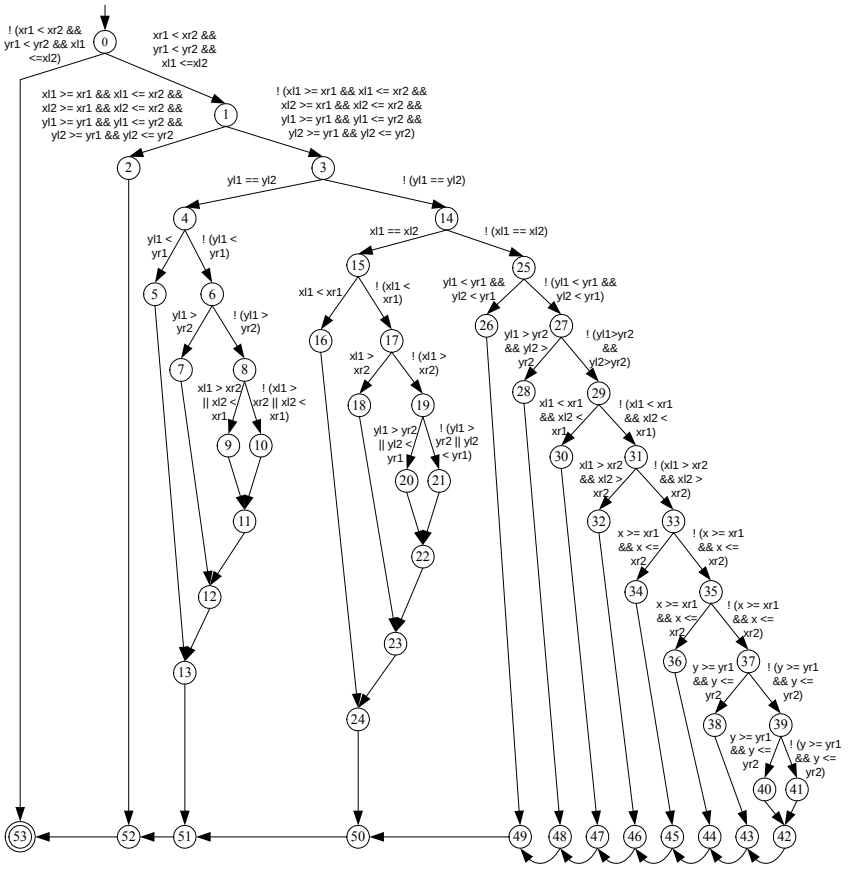

Figura 2. Grafo de control de flujo del programa Position_Line_Rectangle

Los resultados obtenidos por los dos generadores se pueden ver en la Tabla 2. Para cada rango de las variables de entrada se muestra el número de casos de prueba generados, el porcentaje de cobertura alcanzado con dicho número de casos y el tiempo en segundos empleado en ello. TCSS necesita generar menos casos de prueba que TSGen para alcanzar el 100% de cobertura cuando el rango de entrada es de 16 bits, pero cuando dicho rango aumenta TCSS genera más casos de prueba que TSGen. Además TCSS no alcanza el 100% de cobertura en todas las ejecuciones para rangos de 24 bits (2 ejecuciones no logran la cobertura total) y 32 bits (4 ejecuciones no logran la cobertura total), mientras que TSGen

siempre alcanza el 100% de cobertura. Respecto al tiempo consumido por ambos generadores, TCSS consume menos tiempo que TSGen para todos los rangos de entrada.

	Rango: 16 bits (-32768, 32767)			Rango: 24 bits (-8388608, 8388607)			Rango: 32 bits (-2147483648, 2147483648)		
	Casos prueba	% cober-tura	Tiempo (seg.)	Casos prueba	% cober-tura	Tiempo (seg.)	Casos prueba	% cober-tura	Tiempo (seg.)
TCSS	3356	100	1,10	595251	98,3	113,77	858759	97,5	168,34
TSGen	27312	100	57,10	65091	100	147,05	177967	100	454,41

Tabla 2. Resultados obtenidos para TCSS y TSGen

La evolución para cada rango de entrada del número de casos de prueba respecto al porcentaje de cobertura alcanzado se muestra en la Figura 3. TCSS genera menos casos de prueba que TSGen para el rango de 16 bits. Cuando el rango aumenta TCSS genera menos casos de prueba hasta un porcentaje de cobertura del 70%. A partir de ese punto TSGen genera menos casos de prueba.

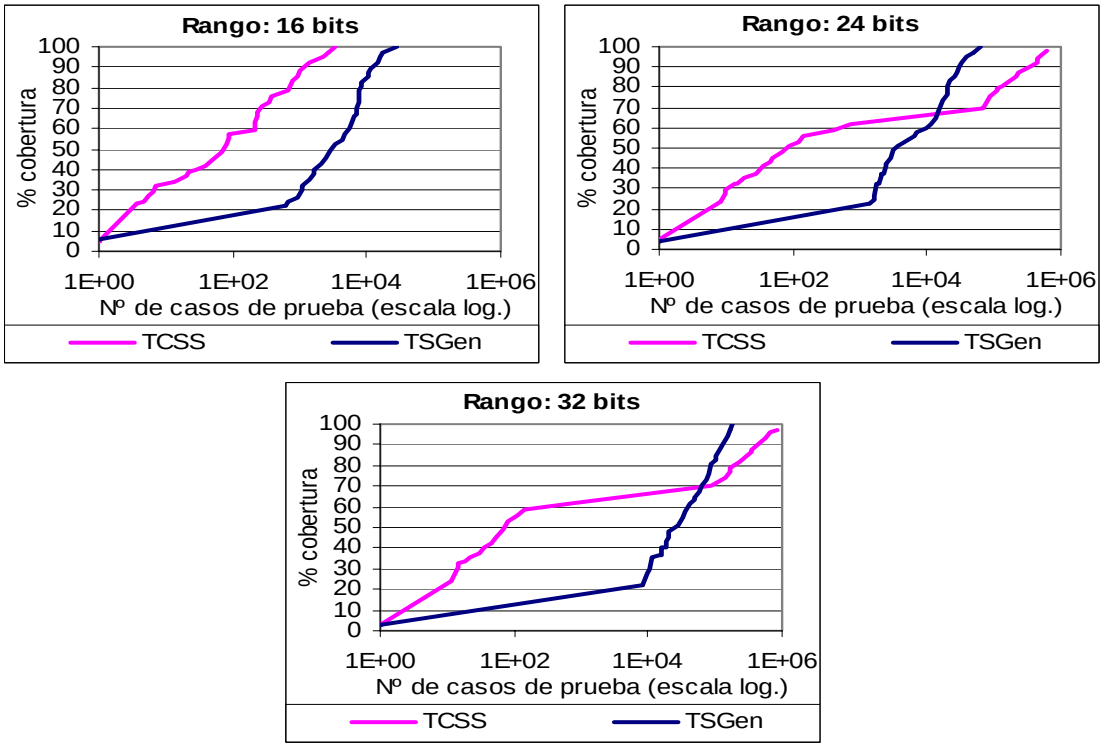

Figura 3. Número de casos de prueba respecto al porcentaje de cobertura alcanzado para cada rango de las variables de entrada

La evolución para cada rango de entrada del tiempo empleado en alcanzar cada porcentaje de cobertura se muestra en la Figura 4. TCSS necesita consumir menos tiempo que TSGen para obtener cada porcentaje de cobertura. A medida que el rango de entrada aumenta, la diferencia de tiempos se hace menor a partir de un porcentaje de cobertura del 70%.

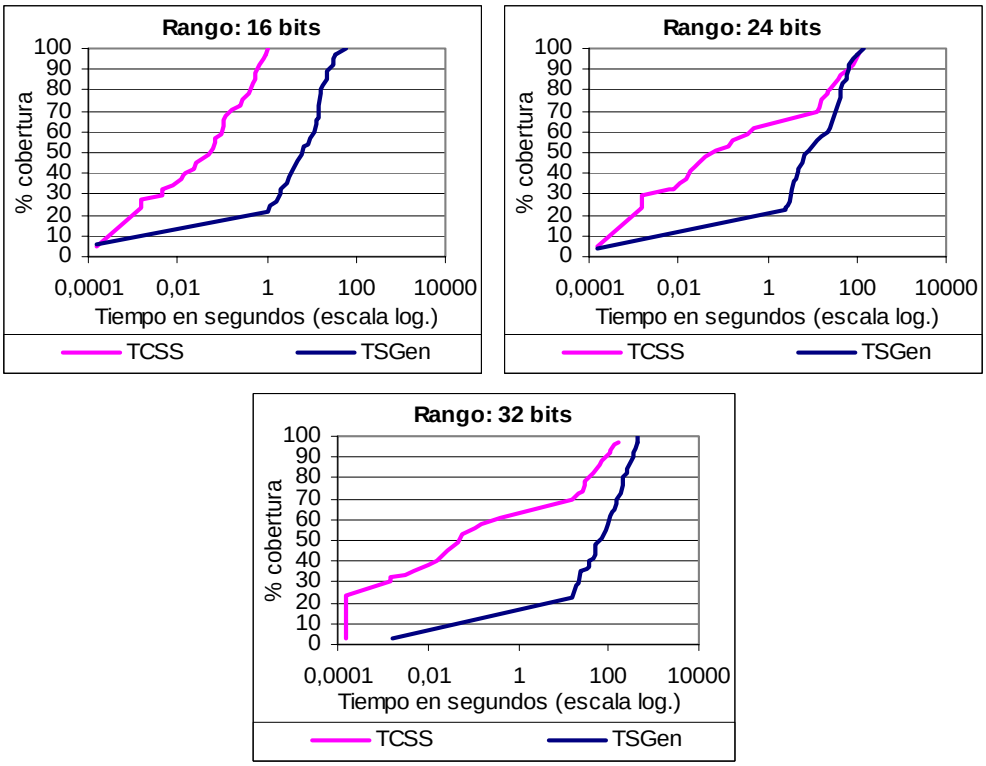

Figura 4. Tiempo empleado respecto al porcentaje de cobertura alcanzado para cada rango de las variables de entrada

5 Conclusiones

En este artículo se ha descrito la adaptación realizada de la técnica metaheurística Búsqueda Dispersa a la generación automática de casos de prueba para cobertura de ramas, dando como resultado el desarrollo del generador de casos de prueba TCSS. Este generador utiliza el grafo de control de flujo asociado al programa bajo prueba y maneja un conjunto de soluciones en cada nodo, que facilita la división del objetivo general de obtener la máxima cobertura de ramas posible en subobjetivos.

Los resultados obtenidos en los experimentos muestran que TCSS se comporta mejor que TSGen con rangos pequeños y en las primeras iteraciones de la búsqueda, debido a la

utilización de la función de diversidad, ya que con esta función TCSS trata de generar casos de prueba que cubran diversas ramas a partir de las soluciones de un determinado nodo. En la cobertura de los nodos más difíciles TSGen se comporta mejor, gracias a la búsqueda local que implementa.

Por lo tanto los resultados sugieren una línea de integración de los generadores TCSS y TSGen aprovechando las mejores características de cada uno, de modo que se puedan mejorar los resultados de ambos.

Actualmente estamos trabajando en la incorporación de una búsqueda local al generador TCSS para mejorar la cobertura de los nodos más difíciles, así como en la utilización de una memoria que permita diferenciar buenas y malas soluciones y la realización de nuevos experimentos que permitan efectuar la comparación de TCSS con otras técnicas de generación automática de casos de prueba.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Educación y Ciencia dentro del Plan Nacional de I+D+I, Proyectos TIN2004-06689-C03-02 (IN2TEST) y TIN2005-24792-E (REPRIS).

Referencias

[1] Beizer B, "Software testing techniques", 2ª edición, Van Nostrand Reinhold, 1990.
[2] Blanco R, Díaz E, Tuya J, "Algoritmo Scatter Search para la generación automática de pruebas de cobertura de ramas", IX Jornadas de Ingeniería del Software y Bases de Datos, pp 375-386, 2004.
[3] Blanco R, Tuya J, Díaz E, Díaz BA, "A Scatter Search approach for automated coverage in software testing", International Conference on Knowledge Engineering and Decision Support, pp 387-394, 2004.
[4] Clarke J, Dolado JJ, Harman M, Hierons RM, Jones B, Lumkin M, Mitchell B, Mancoridis S, Rees K, Roper M, Shepperd M, "Reformulating software engineering as a search problem", IEE Proceedings – Software, 150(3):161-175, 2003.
[5] Díaz E, Tuya J, Blanco R, "Automated Software Testing Using a Metaheuristic Technique Based on Tabu Search", 18th IEEE International Conference on Automated Software Engineering. IEEE Computer Society Press, pp 310-313, 2003.
[6] Díaz E, Tuya J, Blanco R, "Pruebas automáticas de cobertura de software mediante una herramienta basada en Búsqueda Tabú". VIII Jornadas de Ingeniería del Software y Bases de Datos, pp 283-292, 2003.
[7] Glover F, Laguna M, Martí R, "Fundamentals of Scatter Search and Path Relinking", Control and Cybernetics, 39(3):653-684, 2000.

[8] Jones B, Eyres D, Sthamer H, "A strategy for using genetic algorithms to automate branch and fault-based testing", Computer Journal, 41(2): 98-107, 1998.
[9] Korel B,"Automated software test data generation",IEEE Transactions on Software Engineering,16(8):870-870,1990.
[10] Laguna M, Martí R, "Scatter Search: Methodology and Implementations in C", Kluwer Academic Publishers, Boston, 2002.
[11] McCabe TJ, "A complexity measure", IEEE Transaction Software Engineering, 2(4):308-320, 1976.
[12] Michael C, McGraw G, Schatz M, "Generating Software Test Data by Evolution", IEEE Transactions on Software Engineering, 27(12):1085-1110, 2001.
[13] Mansour N, Salame M, "Data generation for path testing", Software Quality Journal, 12(2):121-136, Kluwer Academic Publishers, 2004
[14] Myers G, "The art of software testing", Ed. John Wiley & Sons, 1979.
[15] Pargas R, Harrold MJ, Peck R, "Test-data generation using genetic algorithms", Software Testing, Verification and Reliability, 9(4): 263-282, 1999.
[16] Tracey N, Clark J, Mander K, "Automated program flaw finding using simulated annealing", International Symposium on software testing and analysis, ACM/SIGSOFT, pp 73-81, 1998.
[17] Wegener J, Baresel A, Sthamer H, "Evolutionary test environment for automatic structural testing", Information & Software Technology, 43(14):841-854, 2001.