Capítulo XV
REVISIÓN SISTEMÁTICA DE LITERATURA: FACTORES QUE AFECTAN EL PROCESO DE PRUEBAS DE SOFTWARE

Darío Enrique Soto Durán - dsoto@tdea.edu.co
Facultad de Ingeniería, Tecnológico de Antioquia - I. U., Medellín, Colombia.
Katherine Villamizar Suaza - kvillamizars@gmail.com
Facultad de Ingeniería, Tecnológico de Antioquia - I. U., Medellín, Colombia.
Juan Camilo Giraldo Mejía - jgiraldo1@tdea.edu.co
Facultad de Ingeniería, Tecnológico de Antioquia - I. U., Medellín, Colombia.
Fabio Alberto Vargas Agudelo - fvargas@tdea.edu.co
Facultad de Ingeniería, Tecnológico de Antioquia - I. U., Medellín, Colombia.
Jovani Alberto Jiménez Builes - jajimen1@unal.edu.co
Facultad de Minas, Universidad Nacional de Colombia (Unal), Medellín, Colombia
Adriana Xiomara Reyes Gamboa - axreyes@elpoli.edu.co
Facultad de Ingeniería, Politécnico Jaime Isaza Cadavid (PJIC), Medellín, Colombia

I. INTRODUCCIÓN

El objetivo de las pruebas de software (PS) es garantizar que el producto cumpla con las funcionalidades requeridas y satisfaga con un nivel de calidad aceptable [1]. Las PS son el proceso que asegura la calidad del producto (grado en el que un conjunto de características cumple con los requisitos) [2], [3]. En consecuencia, las PS representan una revisión final de las especificaciones, el diseño y la codificación [4]. Por lo tanto, se consideran como un proceso paralelo al proceso de desarrollo que permite evaluar, desde diferentes aspectos y momentos, el comportamiento de un sistema o componente [5].

Para alcanzar la calidad de un producto existen variables que inciden en el proceso: el tiempo, los recursos, la infraestructura, la cualificación de los profesionales, la metodología empleada, los requisitos del software, etc. [6]. El propósito de esta revisión bibliográfica es identificar los factores que inciden en la efectividad y la eficiencia de las pruebas [7] y realizar una revisión crítica al respecto, que posibilite la síntesis de la información disponible y sea la base para evidenciar posibles mejoras a los problemas o factores que se presenten en la revisión de literatura, teniendo en cuenta diferentes aspectos [8].

Diversos estudios han presentado factores y riesgos que afectan los procesos en el ciclo de vida del software. Un ejemplo es el del "Top 10 de factores que obstaculizan la mejora de los procesos de verificación y validación en organizaciones intensivas en software" [9]. Otro ejemplo es la exposición de los riesgos en proyectos de software [10]. Sin embargo, estos estudios se centran en el proceso del software mas no en los factores que lo afectan.

El presente artículo se encuentra organizado en cuatro secciones: la primera describe el protocolo que orienta la revisión de la literatura, la segunda corresponde al análisis y, la tercera, a los resultados, donde se relaciona el protocolo con el análisis crítico de las investigaciones seleccionadas. Por último, se presentan las conclusiones.

II. METODOLOGÍA

La revisión de literatura se realizó bajo el protocolo de revisión [11], el cual permite, en el contexto de la ingeniería de software, apoyar la labor de los investigadores con el objetivo de evitar sesgos en la realización de este proceso. El protocolo plantea tres fases:

Planificación: Se refiere a las actividades previas a la revisión, para definir las preguntas de investigación, los criterios de inclusión y exclusión, las fuentes de información, la cadena de búsqueda y los procedimientos de categorización.

Ejecución: Se refiere a la búsqueda y la selección de los estudios, con el fin de extraer de ellos los datos y sintetizarlos.

Resultados: Es la fase final, y tiene por objeto la redacción de los resultados de acuerdo con las preguntas de investigación planteadas. Los resultados derivados de la revisión presentan un análisis crítico de los autores.

III. ANÁLISIS

A. Planificación

Preguntas de investigación y justificación: En la Tabla 1 se presentan las preguntas que se propone resolver con la revisión de la literatura, así como la justificación del porqué se plantean.

Tabla 1. Preguntas de Investigación y Justificación

N.°	Pregunta de investigación	Justificación de la pregunta
PI1	¿Cuándo y dónde fue publicado el estudio?	El tema de investigación es amplio y pertinente en la disciplina de ingeniería de software. Esta pregunta de investigación tiene como propósito conocer y comprender las fuentes de las publicaciones específicas para la temática y cuándo han sido publicadas.
PI2	¿Cuál es el estado de los estudios que existen en el proceso de pruebas?	Existen diferentes niveles de madurez que alcanza un estudio en el ámbito científico. Esta pregunta de investigación se enfoca en identificar el estado en que se encuentra la investigación.
PI3	¿Cuáles son los factores que inciden en las pruebas de software?	De forma similar al interrogante anterior, se formula esta pregunta de investigación para establecer las categorías del proceso de pruebas donde se identifican factores que impactan el proceso.
PI4	¿Qué iniciativas se han elaborado para mitigar las problemáticas detectadas?	Identificar las principales soluciones que potencian el proceso de las pruebas de software. Esta información es útil para evaluar los beneficios reales de los estudios actuales en las pruebas de software, así como para señalar los problemas pendientes y, por tanto, la necesidad de profundizar en la temática.

Criterios de inclusión y exclusión: La selección de criterios está organizada en un criterio de inclusión (CI) y cuatro criterios de exclusión (CE). El criterio de inclusión es: (CI1) El estudio analiza los factores que dificultan el proceso de pruebas. Los criterios de exclusión son: (CE1): El estudio no puede ser un artículo corto. (CE2): El estudio tiene un tiempo de publicación mayor a 15 años. (CE3): La publicación es una versión anterior de otro estudio publicado. (CE4): La publicación es un artículo producto de talleres y tutoriales.

Fuentes de búsqueda: La revisión se enfocó en tres bases de datos electrónicas consideradas las más relevantes por [12]. Son ellas: IEEE Xplore (http://ieeexplore.ieee.org), SpringerLink (http://www.springerlink.com) y Science Direct (http://www.sciencedirect.com).

Palabras clave y cadena de búsqueda: La búsqueda de los estudios se realizó teniendo en cuenta el área de interés de la revisión y construyendo la cadena de búsqueda con las palabras claves. La cadena de búsqueda definida fue: ("Problem" OR "risk" OR "barrier") AND ("software testing" OR "software test" OR "Testing" OR "Test").

Validación: Antes de realizar la búsqueda de la información, el protocolo de revisión fue validado evaluando la pertinencia de las palabras claves y la cadena de búsqueda, con una pequeña muestra de publicaciones.

B. Ejecución

Extracción de la información y síntesis: El proceso de búsqueda consideró publicaciones anteriores al mes de junio de 2016. La última búsqueda se realizó en agosto de 2016. Como resultado de la búsqueda se obtuvo un total de 202 publicaciones detalladas así: IEEE XPLORE (65), SpringerLink (39) y Science Direct (98). En primera instancia, se eliminaron las publicaciones duplicadas que aparecen en más de una fuente, y se obtuvo una reducción aproximada del 20 % (quedaron 162 publicaciones). En segundo lugar, se aplicaron los criterios de selección (criterios de inclusión y exclusión) sobre título, resumen y palabras clave, lo cual resultó en 49 documentos (reducción de aproximadamente el 69 %); en el tercer momento, se aplicaron los criterios de selección teniendo en cuenta el texto completo, y el resultado fue un conjunto de 39 publicaciones (reducción de aproximadamente del 20 %). Finalmente se obtuvieron 39 artículos discriminados así:

Tabla 2. Artículos Seleccionados

N.° de referencia	Título	
[13]	Optimal allocation and control problems for software-testing resources.	IEEE
[14]	Analysis of equipment's software test's problems and the countermeasures.	IEEE
[15]	Literature survey of Ant Colony Optimization in software testing.	IEEE
[16]	Analysis of Problems in Testing Practices.	IEEE
[17]	Optimal testing resource allocation, and sensitivity analysis in software development.	IEEE
[18]	On effectiveness of pairwise methodology for testing network-centric software.	IEEE
[19]	When to migrate software testing to the cloud?	IEEE
[20]	A model of third-party integration testing process management for foundation software platform.	IEEE
[21]	Optimal testing resource allocation, and sensitivity analysis for modular software testing.	IEEE
[22]	Reflective Architecture Based Software Testing Management Model.	IEEE
[23]	Comparative analysis of classical multi-objective evolutionary algorithms and seeding strategies for pairwise testing of Software Product Lines.	IEEE
[24]	Reliable code coverage technique in software testing.	IEEE
[25]	Research and implementation of knowledge management methods in software testing process.	IEEE
[26]	Optimal budget spending for software testing under the condition of nonlinear constraint.	IEEE

N.° de referencia	Título	
[27]	An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use.	SCIENCE DIRECT
[28]	Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the Cuckoo Search algorithm.	SCIENCE DIRECT
[29]	Design and analysis of different alternating variable searches for search-based software testing.	SCIENCE DIRECT
[30]	Investigating software testing and maintenance reports: Case study.	SCIENCE DIRECT
[31]	Quality assurance through rigorous software specification and testing: A case study.	SCIENCE DIRECT
[32]	Software safety testing based on STPA.	SCIENCE DIRECT
[33]	Software test process improvement approaches: A systematic literature review and an industrial case study.	SCIENCE DIRECT
[34]	Combinatorial testing for software: An adaptation of design of experiments.	SCIENCE DIRECT
[35]	Achieving quality on software design through test-driven development.	SCIENCE DIRECT
[36]	Constraint-based testing: An emerging trend in software testing.	SCIENCE DIRECT
[37]	Global software testing under deadline pressure: Vendor-side experiences.	SCIENCE DIRECT
[38]	Testing techniques selection based on ODC fault types and software metrics.	SCIENCE DIRECT
[39]	A survey of software testing practices in Canada.	SCIENCE DIRECT
[40]	Analyzing the relationships between inspections and testing to provide a software testing focus.	SCIENCE DIRECT
[41]	A Model-Driven approach for functional test case generation.	SCIENCE DIRECT
[42]	How are software defects found?	SCIENCE DIRECT
[43]	A characterization schema for software testing techniques.	SPRINGER
[44]	Cooperative software testing and analysis: Advances and challenges.	SPRINGER
[45]	Who tested my software? Testing as an organizationally cross-cutting activity.	SPRINGER
[46]	Risk orientation in software testing processes of small and medium enterprises: An exploratory and comparative study.	SPRINGER
[47]	Advances in test generation for testing software and systems.	SPRINGER
[48]	Mining software defect data to support software testing management.	SPRINGER
[49]	Observations on software testing and its optimization.	SPRINGER
[50]	Design and implementation of software testing cases management system based on J2EE.	SPRINGER
[51]	A model-based framework for classifying and diagnosing usability problems.	SPRINGER

Fuente: Elaboración propia

IV. RESULTADOS

¿Cuándo y dónde fue publicado el estudio? (PI1)

Para dar respuesta a este interrogante, las publicaciones se caracterizaron teniendo en cuenta los siguientes criterios:

- Año de publicación (Fig. 1).
- País de la publicación (Fig. 2).
- Tipo de fuentes donde se encuentra la publicación.

Las 39 publicaciones se pueden visualizar en las gráficas. En la Figura 1 se puede observar que el tema en referencia ha tomado mayor relevancia en los últimos tres años, periodo en el que se presenta el 83% (25 de 39) de las publicaciones.

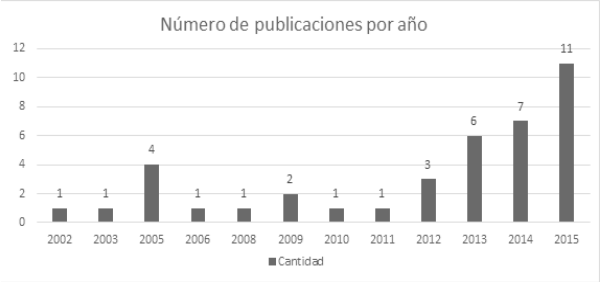

Fig. 1 Cantidad de revisiones por año.

En la Figura 2 se puede concluir que frente a la temática los países con mayor número de publicaciones son EE. UU. y China, con aproximadamente el 51% (20 de 39) del total de estudios seleccionados.

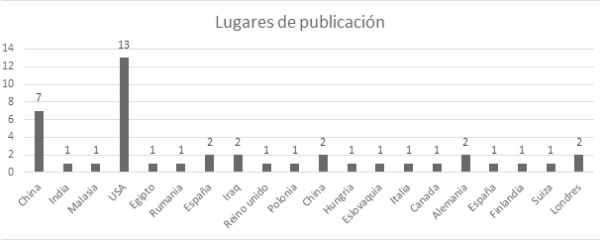

Fig. 2. Cantidad de publicaciones por país

V. DISCUSIÓN

La gestión de conocimiento. Estas categorías clasificadas por año y país agrupan las publicaciones que hacen referencia a la necesidad de incorporar el conocimiento en el proceso de pruebas de software. Diferentes autores plantean como premisa la necesidad de reusar el conocimiento técnico y el conocimiento del dominio de aplicación en los diferentes momentos del proceso con el propósito de optimizar las pruebas. Se retoman soluciones a experiencias previas por ser un soporte para el personal en la toma de decisiones aplicables a diferentes momentos de las pruebas de software. No se pudo comprobar la validación de los estudios analizados.

Gestión del proceso. Esta categoría hace referencia a los diferentes procesos que cubren las actividades de verificación y validación. Se estandariza el marco de procesos en tres niveles: los procesos de organizaciones, los procesos de gestión y los procesos fundamentales.

Diferentes publicaciones identifican problemáticas en el contexto organizacional, específicamente en la estrategia que orienta el proceso, y proponen alternativas que plantean la eficiencia del proceso de pruebas desde la combinación de las técnicas estáticas y dinámicas.

En cuanto a los procesos fundamentales, las publicaciones revisadas analizan los procesos de pruebas desde el enfoque técnico y centran sus esfuerzos en optimizar técnicas de pruebas de caja negra y caja blanca.

Algunas publicaciones hacen un análisis de los criterios que implica la gestión de pruebas en un equipo de pruebas de forma deslocalizada. Este estudio plantea los desafíos y las estrategias que se deben incorporar en una metodología para equipos distribuidos.

Automatización de pruebas. La automatización es una de las alternativas para garantizar la efectividad frente al aumento de la complejidad del software, uno de los desafíos es lograr la automatización del proceso de pruebas al 100 %, y se evidencia que la mayor tendencia de esta clasificación es la generación automática de los casos de prueba.

En consecuencia, se han definido dos subcategorías: la primera denominada "Automatización de procesos", que pretende garantizar el marco metodológico que orienta las organizaciones en las pruebas de software. En segundo lugar se propone la subcategoría "Automatización de pruebas dinámicas". Estos trabajos están enfocados en la automatización de las pruebas funcionales, rendimiento, desempeño y carga.

En el análisis de la revisión se identifican iniciativas que plantean actividades de mitigación a las problemáticas que inciden en las PS y contribuyen a dos aspectos importantes: la reducción de costos y el incremento de la efectividad del proceso.

En la reducción de costos, algunos artículos proponen modelos para optimizar la asignación de recursos a las actividades de pruebas, parten del concepto de confiabilidad para apoyar a los gerentes de proyectos en la toma de decisiones respecto de la definición de recursos del proceso de pruebas.

Alineados con el propósito de incremento de la efectividad del proceso de pruebas, se identifican los siguientes trabajos: La efectividad se plantea a partir de la optimización de los procesos técnicos o fundamentales de las pruebas de software, algunos trabajos presentan investigaciones validadas desde el enfoque de la optimización de las pruebas combinatorias, usadas para detectar con eficacia fallos en el software en las pruebas de caja blanca y caja negra. Los estudios se centran en la generación de casos y datos de pruebas, y usan diferentes enfoques, como: optimización basada en el algoritmo de optimización por colonia de hormigas; lógica difusa, y algoritmos de búsqueda con el propósito de optimizar la eficiencia y el rendimiento de los conjuntos combinatorios de las técnicas de pruebas. En la misma línea se encuentran las investigaciones orientadas a la automatización de las PS focalizadas en los procesos técnicos, con el objeto de aportar a la generación automática de datos de entrada y así validar los casos de prueba, con mayor relevancia para las pruebas de caja negra.

VI. CONCLUSIONES

De acuerdo al análisis de los estudios realizados, se identifican tres factores fundamentales que impactan en un alto nivel el proceso de pruebas como son: el conocimiento, el proceso y la automatización de las pruebas. Siendo el conocimiento un insumo esencial que incide en la efectividad del proceso y la eficiencia de la automatización de las pruebas.

En la dimensión de la gestión de conocimiento (GC), los principales factores identificados que inciden en el proceso de pruebas son:

- Barreras en la transferencia del conocimiento
- Pérdida de conocimiento
- Bajo índice de reúso del conocimiento
- Distribución y uso del conocimiento
- No se consideran de forma adecuada las experiencias y conocimientos adquiridos en la etapa de planificación.

El principal problema de la GC es el reúso del conocimiento, dado que las personas son reticentes para compartir el conocimiento, pues puede convertirse en un elemento competitivo en su entorno laboral, y llega a ser una causa común de varios autores para que la estrategia de GC no logre el objetivo del aprendizaje organizacional.

Dentro de las iniciativas en GC, la reutilización de casos de prueba es la estrategia de mayor prevalencia. La gran mayoría de investigaciones se enfocan en aspectos relacionados con la implementación de los procesos de GC en su conjunto o específicamente en una de sus fases. Por otra parte, los estudios se centran en presentar el diseño de sistemas basados en conocimiento como alternativa de implementación de la GC, y dan solución sólo a un aspecto del proceso de pruebas sin lograr la sinergia entre los diferentes elementos que conforman el sistema, como son las personas, la tecnología y la estrategia organizacional. Las técnicas de GC más utilizadas y con mejores resultados son las páginas amarillas y los mapas de conocimiento. Otras líneas de investigación sin explorar son: la gestión de conocimiento asociado a cada tipo de prueba y el proceso para obtener de manera fiable y estructurada las lecciones aprendidas para su posterior reúso.

En cuanto a la gestión del proceso, los factores de mayor relevancia que impactan el proceso de pruebas son:

- Falta de compromiso de la alta dirección.
- Modelo y metodologías existentes rígidos y complejos de implementar en las organizaciones de software.
- Ausencia o baja calidad de los artefactos de desarrollo.

El compromiso de la alta dirección es un elemento esencial en cualquier iniciativa empresarial de mejora de procesos, sin embargo, el tamaño de la organización y la estructura del equipo de pruebas inciden en la efectividad del proceso. Por ejemplo, en las organizaciones pequeñas, para minimizar los costos de los proyectos, asignan las responsabilidades de pruebas a los desarrolladores de software y soslayan en gran medida los criterios de pruebas. Una alternativa para este tipo de escenarios son los desarrollos basados en metodologías ágiles, porque permiten minimizar riesgos a medida que avanza el desarrollo.

La metodología o el modelo para el desarrollo del proceso de pruebas son un factor que asegura en gran medida el éxito del proyecto, alineados a la política y estrategia de pruebas establecidas en los proyectos de pruebas. En consecuencia, se han construido iniciativas que adoptan metodologías de desarrollo y mejora de procesos acorde a los contextos organizacionales. Con el estándar ISO 29119 recientemente liberado, se unifican conceptos y criterios de la aplicación de pruebas que resuelven el interrogante ¿qué se debe hacer?, pero sin resolver ¿el cómo se hace?, y es una oportunidad para un dominio común que permita establecer procesos para soportar investigaciones en el campo de la mejora de procesos y la gestión de conocimiento.

Otro elemento en esta dimensión es la inadecuada planeación del proceso, focalizada en la dificultad para identificar los riesgos que genera una selección inadecuada de la estrategia. Los activos del proceso de construcción son insumos primordiales para el proceso de pruebas de software, pero en algunos casos resultan incompletos y defectuosos y afectan la ejecución de las pruebas. Esta situación es reiterativa en la tercerización de pruebas y diferentes autores proponen alternativas como: trabajo cooperativo entre desarrolladores y probadores, estudios en análisis de riesgos, reúso de experiencias y optimización de pruebas exploratorias. Sin embargo, se deben explorar procesos de mejora basados en conocimiento y experiencias que permitan estimar y planear con información histórica.

En esta dimensión, las automatizaciones de las pruebas tienen varios factores que se repiten en la causa expresada por las investigaciones analizadas. Veamos:

- Alto nivel de complejidad y bajo nivel de usabilidad en las herramientas de automatización.
- Desconocimiento del uso de las herramientas por parte del personal.
- Falta de estandarización del diseño arquitectónico.
- Recursos ineficientes.

En la actualidad, el proceso de pruebas en los proyectos de desarrollo de software es una etapa sacrificada en tiempo, costos y personal frente a retrasos de desarrollo, que convierte la automatización en un proceso manual sin impacto en la calidad del producto.

Las investigaciones en esta área se enfocan en resolver la automatización de procesos técnicos para mitigar sesgos derivados de las operaciones manuales.

En esta área se pueden identificar nuevas líneas de investigación: la automatización de pruebas en el contexto SOA, líneas de productos y proyectos de explotación de datos.

REFERENCIAS

[1] C. Kaner, B. Johnson, J. Falk y H. Q. Nguyen, “This is near-final, B. L. Recruiting Software Testers”, Software Development, N.° 7, pp. 62-64, 1999, San Francisco,.

[2] ISO, I. IEEE, Systems and Software Engineering--Vocabulary. ISO/IEC/IEEE 24765: (E) Piscataway, NJ: IEEE Computer Society, Tech. Rep, 2010.

[3] C. García, A. Dávila y M. Pessoa, “Test process models: Systematic literature review”, en International Conference on Software Process Improvement and Capability Determination, Springer International Publishing, 2014, pp. 84-93.

[4] F. A. Amo, L. M. Normand, y F. J. S. Pérez, Introducción a la ingeniería del software, Delta Publicaciones, 2005.

[5] G. J. Myers, C. Sandler y T. Badgett, The art of software testing, John Wiley & Sons, 2011.

[6] P. Kruchten, The rational unified process: an introduction, Addison-Wesley Professional, 2004

[7] A. Bertolino, “Software testing research: Achievements, challenges, dreams”, en 2007

Future of Software Engineering, IEEE Computer Society, 2007, pp. 85-103.
[8] M. J. Harrold, "Testing: a roadmap", en Proceedings of the Conference on the Future of Software Engineering, ACM, 2000, pp. 61-72.
[9] J. García, A. de Amescua y M. Velasco, "Top 10 de factores que obstaculizan la mejora de los procesos de verificación y validación en organizaciones intensivas en software", REICIS. Revista Española de Innovación, Calidad e Ingeniería del Software, vol. 2, N.° 2, pp. 18-28, 2006.
[10] L. M. I. Infante, D. E. A. Rabí y M. M. Díaz, "Exposición a los riesgos en un proyecto de software: aplicación del modelo Mogeri", Serie Científica de la Universidad de las Ciencias Informáticas, vol. 6, N.° 6, 2013.
[11] A. Kitchenham, Guidelines for performing systematic literature reviews in software engineering, Technical report, EBSE Technical Report. EBSE, Durham: University of Durham, 2007.
[12] D. H. Ham, "A model-based framework for classifying and diagnosing usability problems", Cognition, technology & work, vol. 16, N.° 3, pp. 373-388, 2014.
[13] H. Ohtera y S. Yamada, "Optimal allocation and control problems for software-testing resources", IEEE Transactions on Reliability, vol. 39, N.° 2, pp. 171-176, 1990.
[14] J. P. Zhang, W. F. Li, J. Yang y Z. Y. Ding, "Notice of Retraction Analysis of equipment's software test's problems and the countermeasures", en International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering, (QR2MSE), IEEE, 2013, pp. 1143-1146.
[15] B. Suri y S. Singhal, "Literature survey of ant colony optimization in software testing", en CSI Sixth International Conference on Software Engineering (CONSEG), IEEE, 2012, pp. 1-7.
[16] J. Kasurinen, O. Taipale y K. Smolander, "Analysis of problems in testing practices", en Software Engineering Conference. APSEC'09, Asia-Pacific, IEEE, 2009, pp. 309-315.
[17] C. Y. Huang y M. R. Lyu, "Optimal testing resource allocation, and sensitivity analysis in software development", IEEE Transactions on Reliability, vol. 54, N.° 4, pp. 592-603, 2005.
[18] K. Z. Bell y M. A. Vouk, "On effectiveness of pairwise methodology for testing network-centric software", en Third International Conference on Information and Communications Technology ITI. Enabling Technologies for the New Knowledge Society, IEEE, December 2005, pp. 221-235.
[19] T. Parveen y S. Tilley, "When to migrate software testing to the cloud?", En Third International Conference on Software Testing, Verification and Validation Workshops (ICSTW), IEEE, 2010, pp. 424-427.
[20] J. Gao, Y. Lan y M. Jin, "A Model of Third-Party Integration Testing Process Management for Foundation Software Platform", en Fourth International Conference on Wireless Communications, Networking and Mobile Computing. WiCOM'08, IEEE, 2008, pp. 1-4.
[21] C. Y. Huang, J. H. Lo, J. W. Lin, C. C. Sue y C. T. Lin, "Optimal resource allocation and sensitivity analysis for modular software testing", en Fifth International Symposium on Multimedia Software Engineering. Proceedings, IEEE, 2003, pp. 231-238.
[22] Y. A. O. Jun-feng, Y. I. N. G. Shi, L. U. O. Ju-bo, X. Dan y J. Xiang-yang, "Reflective architecture based software testing management model", en IEEE International Conference on Management of Innovation and Technology, vol. 2, IEEE, 2006, pp. 821-825.
[23] R. E. Lopez-Herrejon, J. Ferrer, F. Chicano, A. Egyed y E. Alba, "Comparative analysis of classical multi-objective evolutionary algorithms and seeding strategies for pairwise testing of software product lines", en IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 387-396.
[24] D. N. Rao, M. V. Srinath y P. H. Bala, "Reliable code coverage technique in software testing", en International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME), IEEE, 2013, pp. 157-163.
[25] L. Xue-Mei, G. Guochang, L. Yong-Po y W. Ji, "Research and implementation of knowledge management methods in software testing process", en WRI World Congress on Computer Science and Information Engineering, vol. 7, IEEE, 2009, pp. 739-743.
[26] H. Yongming, W. Xianglin y Y. Chaoyuan, "Optimal budget spending for software testing under the condition of nonlinear constraint", Journal of Systems Engineering and Electronics, vol. 14, N.° 3, pp. 92-97, 2003.
[27] T. Mahmoud y B. S. Ahmed, "An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use", Expert Systems with Applications,

vol. 42, N.° 22, pp. 8753-8765, 2015.
[28] B. S. Ahmed, T. S. Abdulsamad y M. Y Potrus, "Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the cuckoo search algorithm", Information and Software Technology, vol. 66, pp. 13-29, 2015.
[29] J. Kempka, P. McMinn y D. Sudholt, "Design and analysis of different alternating variable searches for search-based software testing", Theoretical Computer Science, vol. 605, pp. 1-20, 2015.
[30] P. Janczarek y J. Sosnowski, "Investigating software testing and maintenance reports: Case study", Information and Software Technology, N.° 58, pp. 272-288, 2015.
[31] L. Lin, J. He, Y. Zhang y F. Song, "Quality assurance through rigorous software specification and testing: A case study", Procedia Computer Science, N.° 62, pp. 257-265, 2015.
[32] C. Yang, "Software Safety Testing based on STPA", Procedia Engineering, N.° 80, pp. 399-406, 2014.
[33] W. Afzal, S. Alone, K. Glocksien y R Torkar, "Software test process improvement approaches: A systematic literature review and an industrial case study", Journal of Systems and Software, N.° 111, pp. 1-33, 2016.
[34] R. N. Kacker, D. R. Kuhn, Y. Lei y J. F. Lawrence, "Combinatorial testing for software: An adaptation of design of experiments", Measurement, vol. 46, N.° 9, pp. 3745-3752, 2013.
[35] E. Guerra y M. Aniche, "Achieving quality on software design through test-driven development", en Software Quality Assurance: Large Scale and Complex Software-intensive Systems, Chapter 9, 2015.
[36] A. Gotlieb, "Constraint-based testing: An emerging trend in software testing", Advances in Computers, N.° 99, pp. 67-101, Chapter Two, 2015.
[37] H. Shah, M. J. Harrold y S. Sinha, "Global software testing under deadline pressure: Vendor-side experiences", Information and Software Technology, vol. 56, N.° 1, pp. 6-19, 2014.
[38] D. Cotroneo, R. Pietrantuono y S. Russo, "Testing techniques selection based on ODC fault types and software metrics", Journal of Systems and Software, vol. 86, N.° 6, pp. 1613-1637, 2013.
[39] V. Garousi y J. A. Zhi, A survey of software testing practices in Canada, Journal of Systems and Software, vol. 86 N.° 5, pp. 1354-1376, 2013.
[40] F. Elberzhager, J. Münch, y D. Assmann, "Analyzing the relationships between inspections and testing to provide a software testing focus", Information and Software Technology, vol. 56, N.° 7, pp. 793-806, 2014.
[41] J. J. Gutiérrez, M. J., Escalona y M. Mejías, "A Model-Driven approach for functional test case generation", Journal of Systems and Software, N.° 109, pp. 214-228, 2015.
[42] M. V., Mäntylä y J. Itkonen, "How are software defects found? The role of implicit defect detection, individual responsibility, documents, and knowledge", Information and Software Technology, vol. 56, N.° 12, pp. 1597-1612, 2014.
[43] S. Vegas y V. Basili, "A characterization schema for software testing techniques", Empirical Software Engineering, vol. 10, N.° 4, pp. 437-466, 2005.
[44] T. Xie, L. Zhang, X. Xiao, Y. F. Xiong y D. Hao, "Cooperative software testing and analysis: Advances and challenges", Journal of Computer Science and Technology, vol. 29, N.° 4, pp. 713-723, 2014.
[45] M. V. Mäntylä, J. Itkonen y J. Iivonen, "Who tested my software? Testing as an organizationally cross-cutting activity", Software Quality Journal, vol. 20, N.° 1, pp. 145-172, 2012.
[46] M. Felderer y R. Ramler, "Risk orientation in software testing processes of small and medium enterprises: An exploratory and comparative study", Software Quality Journal, vol. 24, N.° 3, pp. 519-548, 2016.
[47] H. Yenigün, C. Yilmaz y A. Ulrich, "Advances in test generation for testing software and systems", International Journal on Software Tools for Technology Transfer, vol. 18, N.° 3, pp. 245-249, 2016.
[48] R. Hewett, "Mining software defect data to support software testing management", Applied Intelligence, vol. 34, N.° 2, pp. 245-257, 2011.
[49] D. Liu, S. Xu y H. Liu, "Observations on Software Testing and Its Optimization", en Computer and Information Science, R. Lee, ed., Switzerland: Springer International Publishing, 2013, pp. 33-49.
[50] Lu, J., Su, N., & Zhao, A. "Design and implementation of software testing cases management system based on J2EE", en Information Engineering and Applications, Springer London, 2012, pp. 1268-1274.
[51] D. H. Ham, "A model-based framework for classifying and diagnosing usability problems", Cognition, Technology & Work, vol. 16, N.° 3, pp. 373-388, 2014.