Artículos de reflexión

http://dx.doi.org/10.16925/in.v12i20.1482

Análisis del proceso de pruebas de calidad de *software*

Julián Andrés Mera-Paz [1]

[1] *MsC. en Dirección Estratégica de Telecomunicaciones. Docente e investigador de la Facultad de Ingeniería de Sistemas de la Universidad Cooperativa de Colombia, sede Popayán, Colombia*
Correo electrónico: julian_mera_paz@campusucc.edu.co

Recibido: 20 de noviembre del 2015 **Aprobado:** 19 de julio del 2016

Cómo citar este artículo: J. A. Mera-Paz, "Análisis del proceso de pruebas de calidad de *software*", *Ingeniería Solidaria*, vol. 12, no. 20, pp. 163-176, oct. 2016. doi: http://dx.doi.org/10.16925/in.v12i20.1482

Resumen. *Introducción*: este artículo es producto de la lectura, revisión y análisis de libros, revistas y artículos reconocidos por su calidad científica e investigativa que han abordado el proceso de pruebas de calidad de *software*. El autor, con base en su experiencia laboral en empresas de desarrollo de *software*, docencia y otras, ha recopilado y seleccionado información para argumentar y sustentar el porqué de la importancia del proceso de pruebas de calidad de *software*. *Metodología*: se analizó la literatura existente sobre el proceso de pruebas de calidad de *software* en un contexto local, nacional y mundial. Se revisaron de forma exhaustiva las bases de datos ScienceDirect, Elseiver, Springer, Wiley Online Library, proquest, Enginneering Village, Scopus, Dialnet. *Resultados*: se describen conceptos de gran importancia en el proceso de pruebas, características, metodologías y marcos de referencia enfocados a la adecuada implementación de un proceso de pruebas. *Conclusiones*: se generan unos resultados y conclusiones con el fin de que las empresas de desarrollo de productos *software* mejoren el rendimiento y la efectividad, así como la optimización de los procesos de pruebas de calidad de *software*, que además es base fundamental para iniciar procesos de investigación en calidad de *software*.

Palabras clave: calidad, diseño de prueba, herramientas de prueba, niveles de prueba, *software*, *testing*.

p-ISSN 1900-3102 / e-ISSN 2357-6014

http://dx.doi.org/10.16925/in.v12i20.1482

Software quality testing process analysis

Abstract. *Introduction*: This article is the result of reading, review, analysis of books, magazines and articles well known for their scientific and research quality, which have addressed the software quality testing process. The author, based on his work experience in software development companies, teaching and other areas, has compiled and selected information to argue and substantiate the importance of the software quality testing process. *Methodology*: the existing literature on the software quality testing process was analyzed at a local, national and global context. The ScienceDirect, Elsevier, Springer, Wiley Online Library, Proquest, Engineering Village, Scopus, Dialnet databases were thoroughly reviewed. *Results*: important concepts are described in the testing process, characteristics, methodologies and frameworks focused on the proper implementation of a testing process. *Conclusions*: The Article gives rise to results and conclusions in order for software product development companies to improve performance, effectiveness and optimization of software quality testing processes, and it is also a fundamental base to undertake software quality research processes.

Keywords: quality, test design, test tools, test levels, software, testing.

Análise do processo de testes de qualidade de software

Resumo. *Introdução*: este artigo é produto da leitura, revisão, analise de livros, revistas e artigos reconhecidos pela sua qualidade científica e investigativa que trataram do processo de testes de qualidade de software. O autor, baseado em sua experiência trabalhista em companhias de desenvolvimento de software, docência e outras, reuniu e selecionou informações para argumentar e sustentar o porquê da importância do processo de testes de qualidade de software. *Metodologia*: analisou-se a literatura existente sobre o processo de testes de qualidade de software em um contexto local, nacional e mundial. Revisaram-se exaustivamente as bases de dados *ScienceDirect, Elsevier, Springer, Wiley Online Library, Proquest, Enginneering Village, Scopus, Dialnet*. *Resultados*: descrevem-se conceitos de grande importância no processo de testes, características, metodologias e marcos de referência focalizados na adequada implementação de um processo de testes. *Conclusões*: com o artigo geram-se resultados e conclusões visando que as companhias de desenvolvimento de produtos de software melhorem o desempenho, a efetividade e o aprimoramento dos processos de testes de qualidade de software, além disso é a base fundamental para iniciar processos de investigação em qualidade de software.

Palavras chave: qualidade, desenho de teste, ferramentas de teste, níveis de teste, software, *testing*.

1. Introducción

Este artículo se basa en la línea de investigación *Ingeniería de software*, y el tema tratado es el proceso de pruebas de calidad de *software*. En el estado del arte se aborda el proceso de pruebas de calidad de *software*, en el contexto mundial, nacional y local. Además, con este artículo se pretende socializar y dejar un precedente de la importancia que tiene la implementación de un adecuado proceso de pruebas de calidad de *software* en las empresas de desarrollo *software*.

1.1 Antecedentes

Los equipos tecnológicos o electrónicos, en especial las computadoras, hoy en día se utilizan en un gran número de aplicaciones para la vida del ser humano, y en muchas situaciones es crítico el adecuado funcionamiento de las aplicaciones en los equipos tecnológicos. Como ejemplo de esto se pueden mencionar las transacciones electrónicas, negocios de la bolsa de valores, telemedicina o transporte aéreo. Estos son solo algunos casos en los cuales "la tecnología es un factor crítico en las actividades de la vida del ser humano" [1].

Algunos de los ejemplos reales en los cuales el impacto del mal funcionamiento de un *software* ha afectado la vida del ser humano son los siguientes:

- El lanzamiento del cohete Ariane 5, "El vuelo tuvo lugar el 4 de junio de 1996, primer vuelo de la lanzadera Ariane 5 terminó en un fracaso. El fracaso del Ariane 5 fue causado por la pérdida completa de la orientación y la altitud. 37 segundos después del inicio de la secuencia de arranque del motor falló, generando la explosión y destrucción, esta pérdida de información se debió a la especificación y diseño errores en el *software* del sistema de referencia inercial", después del lanzamiento se destruyó la base de lanzamiento, debido a un mal funcionamiento del *software* de control. El Ariane 5 reutilizó las especificaciones del Ariane 4, no se tuvo en cuenta que la trayectoria era distinta y que el código se reutilizó y no se ajustó [2].
- El fracaso de los misiles Patriot: "El 25 de febrero de 1991, durante la Guerra del Golfo, una batería de misiles Patriot estadounidense en Dharan, Arabia Saudita, no logró rastrear e interceptar un misil Scud iraquí entrante. El Scud golpeó un cuartel del ejército estadounidense, matando a 28 soldados e hiriendo a otras 100 personas. Un informe de la oficina de contabilidad general, GAO / IMTEC-92-26, titulado Patriot missile defense: problema de *software* llevado al fracaso del sistema en Dhahran, Arabia Saudí informó sobre la causa de la falla. Resulta que la causa era un cálculo inexacto del tiempo desde el arranque debido a errores aritméticos del ordenador" [3].
- Rayos X letales de la máquina Therac–25: "Therac-25 era una máquina empleada en terapia de radiación, producida por Atomic Energy of Canadá Limited. El problema residía en que, a causa de un error de programación en la interfaz gráfica, se podía dar el caso de enviar la orden de disparar el haz de electrones de alta energía y situar la placa metálica simultáneamente, disparando las partículas antes de que la placa metálica estuviera en posición, exponiendo al paciente a una dosis letal de radiación, más de diez mil rad. Como resultado: al menos seis accidentes entre 1985 y 1987, y que le costó la vida al menos a cinco personas" [4].

El concepto de *pruebas de calidad de software* permite en las empresas con áreas afines a los sistemas, la computación y la informática, brindar productos con altos estándares de calidad y con una disminución de fallos en estos.

2. Estado del arte

A continuación se hace referencia a algunos trabajos, investigaciones e información de proyectos que han sido propuestos o se piensan desarrollar y se relacionan con la temática del artículo.

2.1 En el contexto mundial

En Argentina, el Instituto Nacional de Tecnología Industrial implementó a partir de 2009 el laboratorio de *testing* y aseguramiento de calidad de *software*, que establece como políticas "El testing como parte del proceso de calidad de productos […] Pruebas de calidad a todo tipo de aplicación" [5]. Por su parte, en la Universidad Tecnológica Nacional, Facultad Regional de Buenos Aires, para optar por el título de magíster

en Ingeniería de la Calidad, se encuentra el trabajo titulado "Estudio comparativo de los modelos y estándares de calidad del software" de la licenciada Fernanda Scalone, en el cual se afirma que: "El principal instrumento para garantizar la calidad de las aplicaciones sigue siendo el plan de calidad, el cual se basa en normas o estándares genéricos y en procedimientos particulares. Los procedimientos pueden variar en cada organización, pero lo importante es que estén escritos, personalizados, adaptados a los procesos de la organización y que se sean cumplidos" [6].

En Uruguay, en la Universidad de la República, el trabajo para optar por el título de magíster en Informática, "Proceso de testing funcional independiente", de la estudiante Beatriz Pérez Lamancha, realiza "un estudio del estado del arte en lo referente a las pruebas y al proceso de pruebas de *software*" [7].

En España, en el trabajo para optar por el título doctoral en Ingeniería de Sistemas Telemáticos, titulado "Contribución a la gestión de los procesos de software y servicios", en la Universidad Politécnica de Madrid, del estudiante Hugo Alexer Parada Gelvez, se destaca que "son escasas las investigaciones centradas en desarrollar metodologías y técnicas de pruebas en nuevos dominios" [8].

En Estados Unidos, en el MIT Massachusetts Institute of Technology, en el programa Ciencia, Tecnologia y Sociedad (CTS) [9] se orienta la asignatura Introducción al proceso personal de *software*, y un elemento esencial es el desarrollo de *software*, pero en su libro guía se dedica un capítulo especial al proceso de pruebas de calidad de *software* [10].

En Suecia, en el Instituto de Tecnología Karlskrona y la Universidad de Gotemburgo, se realiza una investigación en la "interactividad de las pruebas de *software* basado en búsquedas bajo un experimento controlado" [11]. Aquí se realiza un proceso de investigación y seguimiento riguroso al sistema de búsqueda interactivo basado en *software* de *testing* (ISBST) a través de un experimento controlado con 58 estudiantes que cursan la asignatura Verificación y validación de *software*. Además, se emplean pruebas manuales y automatizadas, y en este punto se evidencia que las pruebas automatizadas permitieron encontrar mayor número de fallos que los que se obtuvieron en las pruebas manuales, y se decide entre las instituciones reforzar esta asignatura con un laboratorio de seguimiento, verificación y validación de *software*, para lo cual se adecúa infraestructura y se adquieren herramientas para diferentes análisis.

En Beirut, Líbano, en la Universidad Americana de Beirut, los profesores Masri y Zaraket realizaron una investigación titulada "Coverage-Based Software Testing: Beyond Basic Test Requirements" [12], en la cual se analiza la cobertura del *software* basado en pruebas y cómo los requisitos deben ir más allá de una prueba básica; además, se direcciona al lector en la importancia de las pruebas de calidad de *software* desde las etapas tempranas y a profundizar en las pruebas a los requisitos para desarrollar un *software* con calidad.

En Noruega, en la Universidad Noruega de Ciencia y Tecnología, los profesores Deak, Stalhane y Sindre realizaron una investigación denominada "Challenges and strategies for motivating software testing personnel" [13]; en esta, en doce empresas de tecnología en Noruega, seleccionaron a 36 personas para identificar los desafíos y las estrategias para motivar al personal de pruebas de calidad de *software*, y se demuestra que el factor humano, las condiciones y la motivación al talento humano, repercuten en la calidad de los productos *software* que se desarrollan en las empresas.

En Estados Unidos, en la Conferencia internacional de Soft Computing e Ingeniería de Software (scse´15), la ponencia "Quality Assurance through Rigorous Software Specification and Testing: A Case Study" [14] muestra un estudio de caso realizado por profesores de la Universidad Estatal de Ball, Universidad de Purdue y Universidad de Indiana.
En España, en la Universidad de Girona, Beatriz Florian —con el acompañamiento de los profesores colombianos Oswaldo Solarte y Javier Reyes— desarrolló un proceso de investigación denominado "Propuesta para incorporar evaluación y pruebas de usabilidad dentro de un proceso de desarrollo de *software*" [15], en la que se obtiene como resultado que las pruebas y evaluaciones de usabilidad durante el desarrollo del producto *software* han ganado amplia aceptación como estrategia para mejorar la calidad del producto *software*.

En España, en la Universidad de Sevilla, el Ingeniero Rafael Ceballos Guerrero en su tesis doctoral en Lenguajes y Sistemas Informáticos, denominado "Técnicas automáticas para la diagnosis de errores en software diseñado por contrato" [16], asegura que cada vez es más importante la calidad en los productos *software*. Asimismo, resalta que la detección de errores en etapas tardías genera

mayores costos en las etapas de desarrollo de *software* y al final impactan en el precio al cliente; por ende, se enfatiza en utilizar técnicas automáticas para la detección de fallos bajo un proceso de pruebas de calidad de *software*.

2.2 En el contexto nacional

La tesis "Estudio de las prácticas de calidad del *software* implementadas en las Mipymes desarrolladoras de *software* de Pereira", de las estudiantes Paola Andrea Ramírez Aguirre y Carolina Ramírez Arias, para optar por el título de Ingenieras de Sistemas de la Universidad Tecnológica de Pereira, se fundamenta en la siguiente pregunta: "¿Cuál es el grado de implementación de modelos de calidad de *software* en las empresas desarrolladoras de *software* de la ciudad de Pereira?" [17].

En el artículo titulado "Prueba del *software*: más que una fase en el ciclo de vida", se visualiza que "la prueba de *software* es probablemente la parte menos comprendida del ciclo de vida del desarrollo de *software*". Adicionalmente, mediante una propuesta metodológica de cuatro fases, se muestra "por qué es difícil detectar y eliminar errores, por qué es complejo el proceso de realizar pruebas y por qué es necesario prestarle más atención" [18].

En la Fundación Universitaria Tecnológica Comfenalco de Cartagena, la tesis para optar por el título de Ingeniero de Sistemas denominada "Estado del arte de métodos, tipos de testing y herramientas para aplicar pruebas de rendimiento", del estudiante Juan Oliver Navarro [19], se plasman a través de un contenido estructurado, los procesos en el ciclo de desarrollo de *software*, haciendo un énfasis en la importancia de las pruebas de *software* en cada una de las fases.

En la Universidad de Cartagena, la tesis para recibir el título de Ingeniero de Sistemas "Diagnóstico para la implementación de hojas de rutas en la certificación de la industria desarrolladora de *software* en Cartagena de Indias", del estudiante Stalin Oviedo Vargas [20], realiza una investigación con el objetivo de hacer un diagnóstico para implementar hojas de ruta hacia la certificación de calidad en la industria de desarrollo *software* en Cartagena de Indias (Colombia), en enfoques evaluativos de los modelos PSP y TSP, filtrando hacia el modelo CMMI.

2.3 En el contexto local

Son pocos los estudios o documentos que se enfoquen en el proceso de pruebas de calidad de *software*; sin embargo, se encuentran algunos como el realizado en la Institución Universitaria Colegio Mayor del Cauca, trabajo presentado para optar por el título en Tecnólogo en Desarrollo de *Software*, "Propuesta de aplicación de pruebas basada en la norma BS 7925-2 para proyectos de grado enfocados al desarrollo de *software* de la Institución Universitaria Colegio Mayor del Cauca", del estudiante Jhonatan Álvarez Caicedo [21]; y la ponencia presentada en el I Seminario en Calidad de *Software* Mejora de Procesos de Desarrollo, titulada "Mejora de procesos de *software* ágil con Agile SPI Process" [22], de Pardo, Hurtado y Collazos.

3. Metodología

3.1 ¿Por qué son necesarias las pruebas?

Los sistemas de *software* hoy en día son parte importante e integral en nuestras actividades diarias; ejemplo de ello son los cajeros automáticos, los vehículos, los teléfonos inteligentes, las *tablets*, los portátiles, relojes, televisores y muchos otros. Se debe tener en cuenta que los sistemas o aplicaciones son creadas, desarrolladas e implementadas por seres humanos y por ende en cualquiera de sus etapas de creación se puede presentar una equivocación, al generarse esa equivocación se puede llevar a un defecto, como la mala digitación, distracción al codificar, entre otras. "Si no se ha identificado ese defecto y la o las aplicaciones se ejecutan, hay un alto riesgo de que la aplicación no haga lo que debería hacer o el objeto para lo cual fue creada, es decir se genera un fallo o desperfecto" [23].

Es importante conocer que los fallos también se pueden presentar por situaciones del entorno, como la radiación, descarga eléctrica, contaminación, inundaciones, humedad, etc. Las pruebas son necesarias porque con ellas se puede ayudar a reducir los riesgos en las aplicaciones, y lograr de esta manera que se identifiquen los defectos antes de que se ejecuten, con esto se previenen los fallos [24].

3.2 Pruebas

En 1957 se conoce la prueba del *Debugging*[1], y Dijkstra en 1970 presenta una afirmación: "La prueba de *software* puede ser usada para mostrar la presencia de *bugs*, pero nunca su ausencia" [25]. Según Swebook: "Es una actividad realizada para evaluar la calidad del producto y mejorarla, identificando defectos y problemas" [26]. Prueba de *software*: "Es la verificación dinámica del comportamiento de un programa contra el comportamiento esperado, usando un conjunto finito de casos de prueba, seleccionados de manera adecuada" [27].

Según la norma ISO / IEC / IEEE 24765: 2010 se debe tener en cuenta lo siguiente:

"Verificación: Proceso de evaluación de un sistema o componente para determinar si un producto de una determinada fase de desarrollo satisface las condiciones impuestas al inicio de la fase".

"Validación: Proceso de evaluación de un sistema o componente durante o al final del proceso de desarrollo para determinar cuándo se satisfacen los requerimientos especificados". [28]

Con base en estas definiciones, se brinda una respuesta solida respecto a la pregunta: ¿qué es un proceso de pruebas y para qué implementarlo?

3.3 Proceso de pruebas

Muchas empresas de desarrollo de *software* tienen una tendencia a pensar que el proceso de las pruebas de *software* se debe realizar en la última etapa para consolidar la calidad de su producto. Esa tendencia debe dejarse a un lado, ya que las pruebas tienen que estar alineadas al proceso de desarrollo; es clave entonces afirmar que es importante realizar un proceso de pruebas que incluya: "Revisión de los requerimientos, realización de análisis documentales, identificación de defectos, pruebas funcionales y no funcionales, pruebas dinámicas y estáticas, pruebas de integración, informes de confianza en el nivel de calidad, información para la toma de decisiones, planes de mejora continua". [29]

Las actividades de un proceso de pruebas deben cumplir con lo siguiente: planificar y controlar; seleccionar las condiciones de la prueba; diseñar y ejecutar casos de prueba; comprobar los resultados; evaluar los criterios de resultados; elaborar informes del proceso y la aplicación objeto de las pruebas, incluyendo bitácoras de experiencia [30].

Es importante que en el proceso se elabore un plan y una matriz de pruebas para que al ejecutar los casos de prueba se pueda dictaminar si el caso funciona adecuadamente, y así establecerlo como una conformidad; en el evento de que un caso de prueba al ejecutar el producto *software* no funcione de forma adecuada, se relacionará como una no conformidad. A estas métricas se les define un nivel o puntuación acorde a la clasificación, las técnicas y la matriz que se utilicen.

Las métricas ayudan a estimar los tiempos, esfuerzos y asignaciones; determinan también los niveles de riesgo para que el equipo de desarrollo pueda ajustar los flujos de actividades y optimizar así los recursos y el talento de los equipos de desarrollo. Por tanto, la implementación de un proceso de pruebas brinda las pautas para definir objetivos, analizar y viabilizar los requerimientos, diseñar, detallar, programar, implementar y asegurar la calidad de un producto de desarrollo *software*.

3.4 Principios de las pruebas

En los últimos años se ha propuesto una serie de principios que permiten establecer unas pautas comunes para que las empresas de desarrollo de *software* las conozcan y adapten a sus procesos de pruebas [31].

3.4.1 Principio 1. Las pruebas demuestran la presencia de defectos

Las pruebas son unas herramientas que permiten identificar la presencia de defectos; sin embargo, no garantizan que no haya defectos ocultos en el *software*, y el hecho de que no se identifiquen defectos no es una evidencia de que el *software* esté totalmente correcto.

3.4.2 Principio 2. Las pruebas exhaustivas no existen

"Probar todo un aplicativo de extremo a extremo con todas las entradas de datos y condiciones es algo imposible" [32]; en vez de tratar de conseguir

1 *Debugging*: la depuración es un proceso metódico para encontrar y reducir el número de errores o defectos en un programa de ordenador.

ese tipo de pruebas se debe realizar un análisis de los riesgos para establecer prioridades y tomar decisiones adecuadas en la utilización de talento humano y recursos, centralizando los esfuerzos en las pruebas.

3.4.3 Principio 3. Pruebas tempranas

Identificar los defectos en etapas tempranas, cuanto más rápido se identifiquen los defectos, más se ahorrará la empresa en todo tipo de recursos.

3.4.4 Principio 4. Agrupación de defectos

Las pruebas deben concentrarse de manera proporcional. Por lo general, la mayoría de los fallos operativos se concentran en un número reducido de módulos.

3.4.5 Principio 5. Paradoja del pesticida

Si se repite la misma prueba una y otra vez, la misma serie de casos de prueba dejará de encontrar nuevos defectos; es importante que los casos de prueba se revisen periódicamente y escribir nuevos casos de prueba con el objetivo de encontrar más defectos.

3.4.6 Principio 6. Las pruebas dependen del contexto

Las pruebas dependerán del contexto en el cual se ejecuten; por ejemplo, las que sean para un sistema crítico de seguridad como el financiero, se requiere un mayor número de pruebas en comparación con otras aplicaciones de un nivel de complejidad menor o menos críticos.

3.4.7 Principio 7. Falacia de ausencia de errores

La detección y corrección de los defectos no sirven de nada si el sistema no cumple con los requerimientos o necesidades del usuario.

Estos principios que se nombran en diferentes textos o artículos son una guía para tener en cuenta en todo proceso de pruebas de calidad de *software*. El equipo de trabajo debe procurar que se cumplan y tenerlos presentes en cada actuación y ejecución de herramientas o técnicas para brindar aseguramiento a la calidad de un producto *software*.

3.5 Modelo en V en pruebas de calidad de *software*

"En la práctica un Modelo en V puede tener diferentes niveles de desarrollo en función del proyecto y el producto *software*" [33]. Para este caso se mencionan cuatro niveles: prueba de componentes (unidades), de integración, de sistemas y de aceptación.

En la figura 1 se describen las condiciones de la adecuada implementación en este modelo.

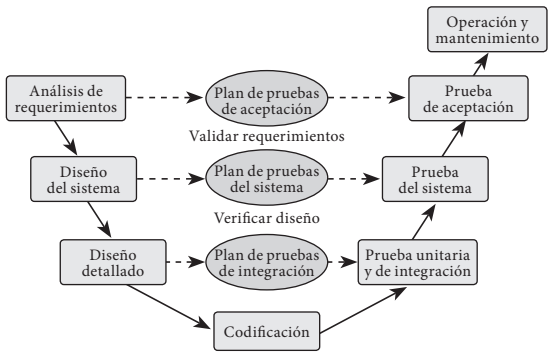

Figura 1. Modelo en V
Fuente: [33]

3.5.1 Pruebas de componentes

Tienen por objeto localizar defectos y comprobar el funcionamiento de módulos *software*, programas, objetos, clases, etc, que puedan probarse por separada. Es decir, se pueden realizar de manera independiente al resto del sistema en función del contexto. Diseño de casos de pruebas: "Requisitos de los componentes, diseño de detalle en los casos de uso, código en el módulo o componente" [29].

3.5.2 Pruebas de integración

Se encargan de probar las interfaces entre los componentes o módulos; por ejemplo, el componente validación de usuario con el sistema operativo, el sistema de archivos en integración con el *hardware*, etc. Diseño de casos de pruebas: "Diseño de *software*, arquitectura, flujos de trabajo, casos de uso, se deben tener en cuenta los objetos de prueba típicos: 1. Base de datos de subsistemas, 2. Infraestructura, 3. Interfaces, 4. Configuración del sistema, 5. Datos de configuración" [27].

3.5.3 Pruebas de sistema

Hacen referencia al sistema como un todo; se debe elaborar un plan de pruebas de forma clara y bien estructurada. Diseño de casos de pruebas: "Requisitos del usuario, requisitos del sistema, casos de uso, procesos de negocio, informes de análisis de riesgo. Se deben tener en cuenta los objetos de prueba típicos: 1. Procesos de negocio en sistema completamente integrado, 2. Procesos operativos y de mantenimiento, 3. Procedimientos de usuario, 4. Formularios, 5. Informes, 6. Datos de configuración". [29]

3.5.4 Pruebas de aceptación

Se enfocan en la aceptación de los criterios previstos en un contrato de desarrollo de *software*, acordado entre la fábrica de *software* y el cliente. "Las pruebas de aceptación del sistema por parte de los administradores del sistema, entre las que se incluyen" [34]:

3.5.5 Pruebas de backup */ restauración*

Recuperación de desastres; gestión de usuarios; tareas de mantenimiento; carga de datos y tareas de migración; comprobaciones periódicas de vulnerabilidades de seguridad; pruebas de aceptación contractual y normativa; pruebas alfa y beta.

Es importante para aplicar el modelo V que se tenga claro por parte de los probadores o *tester* la definición de equivocación, defecto o falta y falla.

En la norma ISO / IEC / IEEE 24765: 2010 se hace referencia a lo siguiente:

"Equivocación (*mistake*): Acción del ser humano que produce un resultado incorrecto".

"Defecto o falta (*fault*): Un paso, proceso o definición de dato incorrecto en un programa de computadora. El resultado de una equivocación potencialmente origina una falla". [28]

"Falla (*failure*): Resultado incorrecto, el resultado de una falta".

3.6 Clasificaciones de las pruebas

Según Whittaker [35], se propone clasificar las pruebas en funcionales, no funcionales y estructurales.

3.6.1 Pruebas funcionales

"Se basan en funciones, prestaciones y en su interoperabilidad con sistemas específicos, y pueden llevarse a cabo en todos los niveles de prueba" [29]. Es importante mencionar que se orientan en el comportamiento externo de un producto o aplicativo *software*, en las pruebas de caja negra.

3.6.2 Pruebas no funcionales

Hacen referencia a las pruebas necesarias "para medir las características de los sistemas y *software* que pueden cuantificarse según una escala variable [36]". Se debe tener en cuenta que se orientan hacia el comportamiento externo del *software* y en la mayoría de los casos utilizan técnicas de diseño de pruebas de caja negra.

3.6.3 Pruebas estructurales

Pueden realizarse en todos los niveles de prueba. Son las más idóneas, después de las técnicas basadas en la especificación, "para ayudar a medir la exhaustividad de las pruebas mediante una evaluación de la cobertura de un tipo de estructura" [29].

Para todo proceso de pruebas se debe tener claridad sobre la diferencia al clasificar los tipos de pruebas, esto contribuye a un análisis sólido del plan de pruebas y a estructurar los casos de pruebas y creación de la respectiva matriz; se tributa además a la eficiencia en el proceso de calidad del producto *software*.

3.7 Técnicas de prueba

Se puede evidenciar en la figura 2 que existen varias técnicas de prueba, que se agrupan de la siguiente manera:

Agrupación	Técnicas
Técnicas de caja negra	Partición de equivalencia Análisis del valor límite Tablas de decisión Máquinas de estado finito Grafo causa efecto Prueba de dominios
Técnicas de caja blanca	Basadas en el flujo de control Basadas en el flujo de los datos Mutantes
Técnicas según quién hace la prueba	Pruebas de aceptación Pruebas alfa y beta Pruebas de usuario Pruebas en pares
Técnicas basadas en la experiencia	Prueba *ad hoc* Conjetura de errores Testing exploratorio

Figura 2. Agrupación y técnicas de pruebas
Fuente: [26]

3.8 Técnicas de caja negra

Partición de equivalencia: "Puede utilizarse para lograr objetivos de cobertura de entrada y salida, con entradas humanas, vía interfaces a un sistema, o parámetros de interfaz de las pruebas de integración" [29]. En esta técnica es importante identificar las clases de equivalencia, por ejemplo, rango de valores entre 1 y 10 serán las clases de equivalencia, es decir que todo valor menor a 1 y todo valor mayor a 10 serán valores inválidos. Luego se generan los casos de prueba con diferentes valores para asegurar que la aplicación solo acepte valores entre 1 y 10.

Análisis de valor límite: "Las condiciones límite son situaciones en los bordes, por arriba, y por debajo de las clases de equivalencia para los valores de la entrada y de la salida" [37]. En esta técnica se identifica la clase de equivalencia válida; por ejemplo, 0 <= J <= 10 000 si 0 es menor o igual que J, el numero – 1 no debería ser tomado por la aplicación, pero el numero 0 es igual a 0, por tanto debería ser un valor válido para la aplicación; si se toma como dato de entrada 10 001 tampoco debería tomarlo la aplicación ya que la variable J debe ser menor o igual que 10 000.

Tabla de decisión: "Las tablas de decisión representan relaciones lógicas entre las condiciones y las acciones. Los casos de prueba son derivados sistemáticamente considerando cada combinación posible de condiciones y acciones" [26]. Para el uso de esta técnica se deben generar condiciones, acciones y reglas para estas.

- Condiciones: todas las situaciones que puedan presentarse.
- Reglas de las condiciones: indican el valor que debe asociarse a cada condición.
- Acciones: lista del conjunto de los pasos por seguir cuando se presente una condición.
- Reglas de acciones: muestran las acciones específicas del conjunto que deben emprenderse dados los valores que toman las condiciones.

Como ejemplo se tiene:

- Condición: cliente de la empresa
- Regla de la condición: sí es cliente – no es cliente
- Acción: aplicar factura con descuento
- Regla de la acción: será las *n* veces que la acción se adapte a la condición

Máquinas de estado finito: "Un sistema puede tener distintas respuestas dependiendo de las condiciones actuales o de su estado. En ese caso, el sistema se puede mostrar como máquinas de estado. Esta forma de modelar permite ver el *software* en términos de sus estados, las transiciones entre los estados, las entradas o los acontecimientos que disparan el cambio de estado y las acciones que pueden resultar de esas transiciones". [29]

Grafo causa efecto: "Ayuda a seleccionar, de una manera sistemática los casos de prueba. Una causa es una condición de entrada o una clase de equivalencia de las condiciones de la entrada. Un efecto es una condición de salida o una transformación del sistema". [43]

Prueba de caso de uso: "Las pruebas pueden derivarse de casos de uso. Un caso de uso describe las interacciones entre los actores, incluyendo usuarios y el sistema" [38]. Los casos de uso se pueden definir a nivel abstracto o a nivel del sistema.

Prueba de dominios: "Un dominio es un conjunto que incluye todos los posibles valores de una variable para una función. En la prueba de dominio se identifican las variables y las funciones. Las variables pueden ser de entrada o de salida. Para cada una, se toman pocos valores representativos de los posibles de la clase de equivalencia (típicamente casos bordes) para cada clase" [39].

3.9 Técnicas de caja blanca

Se basan en una estructura identificada del *software* o del sistema, según unos niveles específicos.

Niveles: "Nivel de componente: estructura de un componente *software*. Ejemplos: sentencias, decisiones, caminos distintos.

Nivel de integración: la estructura se basa en un árbol de llamadas, diagrama en el que un módulo llama a los otros módulos.

Nivel de sistema: la estructura puede ser por menús, ejemplo: proceso de negocio, páginas web" [29].

Basadas en el flujo de control: "Se cubren todas las sentencias o bloques de sentencias en un programa, o combinaciones especificadas de ellas. La adecuación de tales pruebas se mide en porcentajes; por ejemplo, se dice haber alcanzado una cobertura de sentencia del 100 % cuando las sentencias han sido ejecutadas por lo menos una vez por las pruebas" [46].

Basadas en el flujo de los datos: "El criterio más fuerte, all definition-use, requiere que para cada variable, cada segmento de la trayectoria del flujo de control desde una definición de esa variable hasta su uso sea ejecutado. Para reducir el número de las trayectorias requeridas se utilizan estrategias más débiles como all-definitions" [29].

Pruebas mutantes: "Un mutante es una versión levemente modificada del programa a probar, diferenciado de él por un cambio sintáctico pequeño. Puede ser usado para evaluar un conjunto de prueba o criterio de prueba en sí mismo. Para que la técnica sea eficaz, se deben derivar automáticamente de manera sistemática gran cantidad de mutantes" [26].

Técnicas según quien hace la prueba: existen algunas técnicas de prueba que dependen de quién realiza las pruebas; este es el caso de las pruebas de aceptación, las alfa y beta, las de usuario y las pruebas en pares [40].

Pruebas de aceptación: "Es el proceso de comparar el programa contra sus requerimientos iniciales y las necesidades reales de los usuarios. Realizado generalmente por el cliente o el usuario final".

Pruebas alfa y beta: "Antes de que el *software* se libere, se distribuye a un pequeño grupo representativo de los usuarios potenciales para el uso interno (alfa) o externo (beta). Estos usuarios reportan problemas con el producto".

Prueba de usuarios: "Es la prueba realizada por el tipo de persona que usará el producto. Puede ser realizado en cualquier momento durante el desarrollo, en el lugar del cliente o en el de desarrollo, en ejercicios completamente dirigidos o a la discreción del usuario".

Prueba en pares: "Consiste en que dos *tester* trabajan juntos para encontrar errores. Comparten una computadora e intercambian el control de la misma mientras prueban" [26].

Técnicas basadas en la experiencia: existen técnicas de prueba que se basan en la intuición, el conocimiento o la experiencia de la persona que realiza las pruebas.

Pruebas *ad hoc*: "Quizás la técnica más practicada, las pruebas son derivadas confiando en la habilidad, intuición, y experiencia con programas similares. Puede ser útil para identificar pruebas que no son fácilmente encontradas por técnicas más formales" [26].

Conjetura de errores: "La idea básica es enumerar una lista de errores y después escribir los casos de prueba basados en la lista. Otra idea es identificar los casos de prueba asociados a asunciones que el programador pudo haber hecho cuando leía la especificación" [26].

Testing exploratorio: el término fue introducido por Kaner, y se trata de "ejecutar las pruebas a medida que se piensa en ellas, sin gastar demasiado tiempo en preparar o explicar las pruebas, confiando en los instintos. Se define como el aprendizaje, el diseño de casos de prueba y la ejecución de las pruebas en forma simultánea" [40].

4. Resultados

Por la naturaleza del artículo, la discusión de los resultados se fundamenta en referencias bibliográficas, es decir, lo que aparece detallado constituye un marco teórico, que fundamenta una guía teórica para que las empresas de desarrollo de *software* y universidades adapten los conceptos a la implementación del proceso de pruebas. Se debe tener en cuenta que los procesos de pruebas se desarrollan por personas, son susceptibles a errores y por eso se deben reforzar bibliográficamente.

Como lo plantean Banerjee, Chattopadhyay y Roychoudhury [41], en el capítulo 3 del libro *Los avances en informática*: "Para las últimas décadas, los sistemas integrados han ampliado su alcance en los principales aspectos de las vidas humanas. A partir de pequeños dispositivos portátiles (teléfonos inteligentes), como a los sistemas de automoción avanzadas (como los sistemas antibloqueo de frenos), el uso de sistemas embebidos ha aumentado a un ritmo dramático. El *software* integrado se especializó en *software* que se pretende que funcione en dispositivos embebidos. Además, los sistemas embebidos a menudo se requieren para operar en la interacción con el medio físico, la obtención de los componentes a factores ambientales (tales como la temperatura o la presión del aire). La necesidad de interactuar con un entorno físico dinámico, a menudo no determinista, aumenta aún más los problemas asociados con las pruebas y validación de *software* embebido". Se demuestra una vez más que es una necesidad vital enfocarse en los procesos de pruebas de calidad de *software*, ya que los sistemas están en la mayoría de los dispositivos

tecnológicos y muchos de ellos están presentes en las actividades humanas; por ende, el proceso de calidad debe ser sólido.

Las empresas y universidades han descuidado el proceso de pruebas de calidad de *software* y la esencia e importancia de la afectación en las actividades cotidianas. Un adecuado proceso de pruebas de calidad de *software* mitiga la generación de riesgos, más cuando hoy se habla de la implicación de *software* en procesos tan críticos como las cirugías o procedimientos de diagnóstico médico.

Es primordial recordar que los productos de desarrollo *software* y los casos de prueba son elaborados por seres humanos y que se pueden presentar equivocaciones; por lo tanto, es útil conocer la información y el proceso de pruebas de calidad de *software*, para que dentro del ciclo de producción se establezcan las instrucciones, roles y responsabilidades de forma clara.

El equipo de desarrollo o producción debe tener claras las condiciones de funcionabilidad y usabilidad del producto *software*; además, es recomendable realizar comparaciones con otros proyectos o módulos desarrollados, no con el objetivo de medir la efectividad laboral, sino para tener referentes en la estructura, la organización, los tiempos y la calidad.

El director o gerente del proyecto debe estar centrado en que los clientes buscan productos que satisfagan sus necesidades, y la calidad es el factor principal; por eso se tienen que seguir unos principios y clasificaciones, que a través de un plan de pruebas, sean guía para el equipo de desarrollo, y que este realice una optimización de las herramientas, técnicas y conocimientos.

Al cliente como elemento valioso de todo proyecto se le involucra en el proceso de pruebas de calidad, para que esté enterado de los avances, para que conozca los posibles fallos y qué no es un fallo; asimismo, para que tenga los medios para reportar los fallos. Es importante que en esa relación con el cliente el equipo de desarrollo y el gerente del proyecto transmitan seguridad y franqueza, aspecto que permitirá tener un cliente satisfecho y una imagen corporativa sólida.

En este artículo se evidencia que hay un camino por recorrer y gran cantidad de información y temáticas que se derivan del proceso de pruebas de *software*. Como trabajo futuro se plantea el estudio de pruebas de calidad de *software* para el desarrollo de *software* móvil y la implementación de un laboratorio para prácticas de estudio e investigación en la facultad de Ingenierías, de la Universidad Cooperativa de Colombia, sede Popayán.

5. Discusión

Se evidencia que hay un gran número de temáticas y terminología sobre el proceso de pruebas de calidad de *software* pero es poca la valoración que se le está haciendo a la importancia e impacto en las fábricas o empresas de *software* y en la academia. Con este artículo se analiza la importancia del proceso de pruebas de calidad de *software* con la finalidad de que estas apunten a mejorar el rendimiento, efectividad y optimización en la calidad de los productos *software*.

Este artículo tributa a los equipos de desarrollo *software*, con el fin de que se les entregue una valoración importante a las pruebas de calidad y se conozca el estado del producto en cuanto a faltas o defectos encontrados, y una comparación entre las etapas o avances del proyecto o, si es posible, respecto a otros proyectos. De esta manera, se busca consolidar unos niveles altos de calidad y optimizar recursos. Se ayuda al director y otros responsables del proyecto. Además, se puede dimensionar la importancia del proceso de pruebas de calidad, para con ello estructurar de forma adecuada tiempos, roles, responsabilidades y asignaciones que lleven a un *software* de calidad.

Elaborar procesos de pruebas de calidad de *software* sin un orden definido o sin conocimiento previo no es recomendable, ya que va a dificultar la comprobación de la funcionabilidad y usabilidad de un producto de desarrollo *software*. Se podría argumentar que la finalidad de la calidad va a ser el uso y el contexto de este que le brinde el cliente o usuario final, pero ello se puede refutar, ya que quien desarrolla el producto puede tener una concepción diferente y argumentada sobre el funcionamiento de un producto *software*, como el rendimiento, capacidad de almacenamiento, etc.

Como lo mencionaron Mota, Carmo, Mcgregor, Santana y Romero, "en el desarrollo de *software*, la prueba es un mecanismo importante tanto para identificar defectos y asegurar que los productos bajo la conformidad con las

especificaciones. Esta es una práctica común en el desarrollo de un solo sistema" [42]. Así pues, se puede ahondar más en el tema de las pruebas como un factor de identificación de defectos, ya que este artículo apunta a la importancia de pruebas de calidad de *software* con un enfoque hacia el aseguramiento de esa calidad.

Varios de los autores seleccionados en este artículo mencionan, de diferentes formas, que no existe una calidad perfecta o absoluta, y esto se determina por los principios de las pruebas de *software*; por tanto, es recomendable que los equipos de desarrollo sustenten sus pruebas en observaciones dadas por la programación entre pares, las experiencias adquiridas y las apreciaciones del cliente. Con ello se puede acercar a una calidad óptima, adecuada al contexto funcional.

Como especifica Kuhn en su artículo: "Investigamos informes de error de dos proyectos de *software* de código abierto grandes, un navegador y el servidor web, para proporcionar respuestas preliminares a tres preguntas: ¿Hay un punto de rendimientos decrecientes en el que la generación de todas las combinaciones de grado n es casi tan eficaz como las n + combinaciones de 1 grado? ¿Cuál es el valor apropiado de n para clases particulares de *software*? ¿Este valor difiere para diferentes tipos de *software*, y por cuánto? Nuestros hallazgos sugieren que más del 95 % de los errores en el *software* estudiado serían detectados por los casos de prueba que cubren todas las combinaciones de 4 vías de valores, y que el *software* del navegador y el servidor fueron similares en el porcentaje de errores detectables por combinaciones de grado 2 a 6" [43].

Por otro lado, en el caso planteado, se evidencia que un 95 % de los errores fueron detectados gracias a los casos de prueba, lo que respalda que con este artículo se coadyuva a los potenciales clientes y usuarios finales, ya que son estos en realidad los que interactúan con los productos *software*, pues ellos son los que detectan en su mayoría los defectos y pueden dar su percepción de los resultados de las pruebas finales, que ya es la puesta en producción de un producto *software*.

Los docentes, estudiantes, investigadores, empresas de desarrollo *software* y comunidad interesada podrán, a través de este artículo, iniciar procesos de ejercicio académico y de investigación que colaboren con la actualización y mejora continua de un proceso de pruebas de calidad de *software*.

6. Conclusiones

Se observa que el proceso de pruebas impacta en los riesgos del producto *software*; por ende, para las empresas de *software* es vital formular y adecuar un buen proceso de pruebas de calidad de *software*.

La mayoría de las universidades en Latinoamérica han estado aisladas en cuanto a la implementación de pruebas de calidad de *software* y la generación de espacios de laboratorio para el desarrollo de técnicas, métricas, indagaciones e investigación en las áreas de informática, sistemas o afines, que permitan afianzar el aseguramiento de la calidad de los productos *software*.

Este artículo describe un resumen del análisis de la importancia del proceso de pruebas de calidad de *software*, como guía de estudio y análisis para equipos de desarrollo *software*, directores y responsables de proyectos, clientes y usuarios finales, docentes y estudiantes, demás comunidad interesada para implementar, revisar y consolidar un adecuado proceso de pruebas de calidad *software*.

Con la claridad de la importancia del proceso de pruebas de calidad en los productos *software* se eleva la calidad y fiabilidad dentro del ciclo de vida de un proyecto. Asimismo, al implementar un proceso de pruebas de calidad de *software* se genera un control y seguimiento a los defectos o faltas y a los fallos, de manera que las soluciones que se generen tengan un mayor nivel de calidad.

Es importante que los diferentes roles en una empresa de desarrollo *software* o en un equipo de trabajo tengan un conocimiento básico sobre el proceso de pruebas de calidad. Esto permite que en cada etapa del ciclo del proyecto se manejen mejores estándares de producción y por ende de calidad.

En un sentido general, la implementación de un proceso de pruebas de calidad de *software* en las empresas o universidades instituye un gran avance en el intento por garantizar márgenes de calidad en los productos de desarrollo *software*, y es un elemento estratégico para la imagen corporativa o institucional.

Referencias

[1] L. A. Salinas Arreortua, "El desarrollo tecnológico en el contexto de la modernidad" *Scripta Nova*, vol. VIII, no. 170, pp. 26 -29, 1 agosto 2004 [en línea]. Disponible en: http://www.ub.edu/geocrit/sn/sn-170-26.htm

[2] D. N. Arnold, "The Explosion of the Ariane 5", p. 1, 23 agosto 2000 [en línea]. Disponible en: http://www.ima.umn.edu/~arnold/disasters/ariane.html

[3] D. N. Arnold, "The Patriot Missile Failure", p. 1, 23 agosto 2000 [en línea]. Disponible en: http://www.ima.umn.edu/~arnold/disasters/patriot.html

[4] N. Leverson y C. S. Turner, "An Investigation of the Therac-25 Accidents", *IEEE Computer*, vol. 26, no. 7, pp. 18-41, 7 julio 1993 [en línea]. Disponible en: http://courses.cs.vt.edu/~cs3604/lib/Therac_25/Therac_1.html

[5] Instituto Nacional de Tecnología Industrial, "Laboratorio de Testing Córdoba", pp. 1-21, agosto 2015 [en línea]. Disponible en: http://www.inti.gob.ar/software/

[6] F. Scalone, "Estudio comparativo de los modelos y estándares de calidad del software", Tesis de Maestría, Facultad Regional Buenos Aires, Universidad Tecnológica Nacional, Buenos Aires, Argentina, junio 2006 [en línea]. Disponible en: http://laboratorios.fi.uba.ar/lsi/scalone-tesis-maestria-ingenieria-en-calidad.pdf

[7] B. Pérez-Lamancha, "Proceso de testing funcional independiente", Tesis de Maestría, Facultad de Ingeniería, Instituto de Computación (INCO), Montevideo, Uruguay, 2006 [en línea]. Disponible en: http://www.ces.com.uy/documentos/imasd/Tesis-Beatriz_Perez_2006.pdf

[8] H. A. Parada-Gélvez, "Contribución a la gestión de los procesos de software y servicios", Tesis doctoral, Escuela Técnica Superior de Ingenieros de Telecomunicación (UPM), Madrid, España, agosto 2010, [en línea]. Disponible en: http://oa.upm.es/4019/

[9] Massachusetts Institute of Technology, "Programa en Ciencia, Tecnología y Sociedad", junio 2016, Estados Unidos [en línea]. Disponible en: http://web.mit.edu/sts/

[10] W. S. Humphrey, *Introducción al Proceso Software Personal (PSP)*. Madrid, España: Addison Wesley, 2001, p. 328 [en línea]. Disponible en: http://dis.unal.edu.co/~icasta/GGP/_Ver_2012_1/Documentos/psp.pdf

[11] B. Marculescu, S. Poulding, R. Feldt, K. Petersen y R. Torkar, "Tester interactivity makes a difference in search-based software testing: A controlled experiment", en *Information and Software Technology*, vol. 78, pp. 66-82, dic. 2015 [en línea]. doi: 10.1016/j.infsof.2016.05.009

[12] W. Masri, F.A Zaraket, "Coverage-Based Software Testing: Beyond Basic Test Requirements", en *Advances in Computers*, vol. 103, A. Menom Ed. Estados Unidos: Academic Press, 2016, p. 79-142, 2016. doi: 10.1016/bs.adcom.2016.04.003

[13] A. Deak, T. Stalhane y G. Sindre, "Challenges and strategies for motivating software testing personnel", *Information and Software Technology*, vol. 73 pp. 1-15, 2016.
doi: http://dx.doi.org/10.1016/j.infsof.2016.01.002

[14] L. Lin, J. He, Y. Zhang y F. Song, "Quality Assurance through Rigorous Software Specification and Testing: A Case Study", *Procedia Computer Science*, vol. 62, pp. 257-265, 2015. doi: 10.1016/j.procs.2015.08.448

[15] B. E. Florian, O. Solarte, M. J. Reyes-Moreno, "Propuesta para incorporar evaluación y pruebas de usabilidad dentro de un proceso de desarrollo de software", *Revista EIA*, no. 13, pp. 123-141, 2010 [en línea]. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=3274667

[16] R. Ceballos-Guerrero, "Técnicas automáticas para la diagnosis de errores en software diseñado por contrato", Tesis doctoral, E.T.S. Ingeniería Informática, Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, España, 2011 [en línea]. Disponible en: https://dialnet.unirioja.es/servlet/tesis?codigo=24794

[17] P. A. Ramírez-Aguirre y C. Ramírez-Arias, "Estudio de las prácticas de calidad del software implementadas en las Mipymes desarrolladoras de software de Pereira", Tesis de grado, Facultad de Ingenierías Universidad Tecnológica de Pereira, Colombia, 2010 [en línea]. Disponible en: http://repositorio.utp.edu.co/dspace/bitstream/11059/1977/1/0053R173e.pdf

[18] E. Serna y F. Arango, "Prueba del software: más que una fase en el ciclo de vida", *Revista de Ingeniería*, no. 35, pp. 34-40, jul.-dic. 2011 [en línea]. Disponible en: https://ojsrevistaing.uniandes.edu.co/ojs/index.php/revista/article/view/144

[19] J. O. Navarro, "Estado del arte de métodos, tipos de testing y herramientas para aplicar pruebas de rendimiento", Tesis de grado, Fac. Ing. Sist., Fundación Universitaria Tecnológica de Comfenalco, Cartagena, Colombia, 2010.

[20] S. Oviedo, "Diagnóstico para la implementación de hojas de rutas en la certificación de la industria desarrolladora de software en Cartagena de Indias", Tesis de grado, Fac. Cienc. e Ing., Prog. Ing. Sist., Universidad de Cartagena, Colombia, 2013 [en línea]. Disponible en: http://190.242.62.234:8080/jspui/handle/11227/391

[21] J. Álvarez Caicedo, "Propuesta de aplicación de pruebas basada en la norma bs 7925-2 para proyectos de

grado enfocados al desarrollo de software de la institución universitaria colegio mayor del Cauca", Tesis de grado, Prog. Ing. Infor., Institución Universitaria Colegio Mayor del Cauca, Colombia, 2009 [documento impreso].

[22] C. Pardo, J. A. Hurtado y C. A. Collazos, "Mejora de procesos de software ágil con Agile SPI Process", *Revista DYNA,* vol. 77, n.º 1 164, pp. 251-263, dic. 2010. Disponible en: http://www.revistas.unal.edu.co/index.php/dyna/article/view/25595

[23] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams", en *Future of Software Engineering*. L. Briand y A. Wolf, Eds. Washington: IEEE Computer Society, 2007, pp. 85-103 [en línea]. Disponible en: http://dl.acm.org/citation.cfm?id=1254712

[24] B. Hetzel, *The Complete Guide to Software Testing*, Estados Unidos: John Wiley & Sons; 1993.

[25] E. W. Dijkstra, "Notes on Structures Programming", Technical University Eindhoven, The Netherlands, departamento de Matemáticas, Technical Report 70-WSK-03 1970, pp. 15-64 [en línea]. Disponible en: https://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.PDF

[26] IEEE Computer Society, "Guide to the Software Engineering Body of Knowledge Swebok", IEEE Computer Society, pp. 10-40, 2004 [en línea]. Disponible en: https://www.computer.org/web/swebok/v3

[27] International Software Testing Qualifications Board [ISTQB], "Oficinas principals", pp. 28-40, 2011. Bruselas, Bélgica [en línea]. Disponible en: http://www.istqb.org/downloads/category/2-foundation-level-documents.html

[28] IEEE Explore, 24765-Systems and software engineering. Vocabulary, 2010 [en línea]. doi: 10.1109/IEEESTD.2010.5733835

[29] T. Müller, *Libro Programa de Estudio de Nivel Básico para Probador Certificado*, ISTQB, Version 2013, p. 14. Disponible en: http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4

[30] International Software Testing Qualifications Board [ISTQB], "Certified Tester Foundation Level Syllabus. Released version 2013", 2013 [en línea]. Disponible en: http://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html

[31] International Software Testing Qualifications Board [ISTQB], "Certified Tester Foundation Level Syllabus. Released version 2011", 2011 [en línea]. Disponible en: http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4

[32] D. R. Kuhn, D. R. Wallace y A. M. Gallo, "Software Fault Interactions and Implications for Software Testing", en: *IEEE Transactions On Software Engineering*, Vol. 30, no. 6, pp. 418-421, junio 2004 [en línea]. doi: http://doi.ieeecomputersociety.org/10.1109/TSE.2004.24

[33] Sh. Lawrence Pfleeger y J. M. Atlee, *Software Enginnering*. Estados Unidos: Pearson; 2009.

[34] M. G. Merayo y E. Montes de Oca, "Testing Software and Systems", en: *International Conference, ICTSS 2014*, Madrid, España, 23-25, septiembre 2014 [en línea]. Disponible en: http://www.springer.com/la/book/9783662448564#

[35] J. Whittaker, "What is Software Testing? And Why Is It so hard?", *IEEE Software*, vol. 17, no. 1, pp. 70-79, ene.-feb., 2000. doi: 10.1109/52.819971

[36] IEEE, "Software Engineering. Product quality. Part 1: Quality Model", ISO/IEC 9126-1:2001. 2001 [en línea]. Disponible en: http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749

[37] Myers G, T. Badgett y C. Sanders, *The art of software testing*. New Jersey, Estados Unidos: John Wiley & Sons Inc., 2004, p. 254.

[38] I. Jacobson, G. Booch y J. Rumbaugh, *The Unified Software Development Process*". Boston, Estados Unidos: Addison Wesley, 1999, pp. 185-190.

[39] C. Kaner, J. Bach J. y B. Pretichord, *Lessons Learned in Software Testing*, Wiley & Sons Inc, 2001.

[40] C. Kaner, J. Falk y H. Nuyen. *Testing Computer Software*. Nueva York, Estados Unidos: 1999, p. 286. [en línea]. Disponible en: http://www.amazon.com/Testing-Computer-Software-2nd-Kaner/dp/0471358460

[41] A. Banerjee, S. Chattopadhyay y A. Roychoudhury, "On Testing Embedded Software", *Advances in Computers*, vol. 101, pp. 121-153, 2016 [en línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0065245815000662. doi: 10.1016/bs.adcom.2015.11.005

[42] P. A. da Mota Silveira Neto, I. do Carmo Machado, J. D. McGregor, E. Santana de Almeida y S. R. de Lemos Meira, "A Systematic Mapping Study of Software Product Lines Testing", *Information and Software Technology*, vol. 53, no. 5, pp. 407-423, may. 2011 [en línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0950584910002193. doi: 10.1016/j.infsof.2010.12.003

[43] D. R. Kuhn y M. J. Reilly, "Una investigación de la aplicabilidad de diseño de experimentos para pruebas de software", *Software Ingeniería Taller. Actas 27, Anual de la NASA Goddard / IEEE,* 2002, pp. 91-95.