Revista Cubana de Ciencias Informáticas
Vol. 14, No. 3, Mes Julio-Septiembre, 2020
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 83-104

Tipo de artículo: Artículo original

Temática: Pruebas de software

Recibido: 01/04/2020 | Aceptado: 11/05/2020

Estrategia de pruebas para organizaciones desarrolladoras de software

Testing strategy for software development organizations

Aymara Marin Diaz[1*] https://orcid.org/0000-0001-5101-7804
Yaimí Trujillo Casañola[1] https://orcid.org/0000-0002-3138-011x
Denys Buedo Hidalgo[1] https://orcid.org/0000-0002-5031-2040

[1]Universidad de las Ciencias Informáticas, Cuba. Carretera San Antonio Km 2 1/2. {amarin, yaimi, dbuedo}@uci.cu

*Autor para la correspondencia. (amarin@uci.cu)

RESUMEN

La realización de pruebas para evaluar el software se ha convertido en una tendencia en la industria, pero disimiles investigaciones y tendencias evidencian que se realizan luego de finalizado el producto y muchas veces solo se ejecutan pruebas funcionales. Esto supone un problema pues se aleja la detección del defecto del momento en que se introduce, lo que incide en los costos de corrección y alargan los cronogramas del proyecto, sin cubrir las pruebas

estructurales y no funcionales. En el presente artículo describe una estrategia de pruebas que abarca pruebas funcionales, no funcionales, estructurales y asociadas al cambio, es independiente del negocio, del tipo de producto y de la metodología de desarrollo de software. La estrategia tiene en cuenta buenas prácticas documentas en modelos, normas y estándares reconocidos internacionalmente, que a su vez fueron enriquecidas y particularizadas por expertos de organizaciones cubanas. Se integra el qué probar a partir del tipo de prueba por niveles y el cómo probar a partir de describir los objetivos, objetos de pruebas típicos, bases de pruebas, enfoques y responsabilidades, defectos y tipos de fallas típicos y técnicas y estrategias. Se muestran los resultados de la valoración de expertos y un estudio de casos para mejor comprensión de la propuesta.

Palabras clave: calidad; defectos; estrategia; pruebas; software.

ABSTRACT

Testing to evaluate software has become a trend in the industry but dissimilar investigations and trends show that they are carried out after the product is finished and often only functional tests are executed. This is a problem since the detection of the defect is moved away from the moment it is introduced, which affects the correction costs and lengthens the project schedules, without covering the structural and non-functional tests. This article describes a testing strategy that encompasses functional, non-functional, structural and change-associated tests, independent of the business, the type of product and the software development methodology. The strategy takes into account good documented practices in internationally recognized models, norms and standards, which in turn were enriched and individualized by experts from Cuban organizations. It integrates what to test from the level test type and how to test from describing the objectives, typical test objects, test bases, approaches and responsibilities, defects and typical failure types

and techniques and strategies. The results of the expert assessment and a case study are shown to better understand the proposal.

Keywords: quality; defects; strategy; test; software.

Introducción

La presencia del software se incrementa de manera sostenida en innumerables actividades del ser humano, entre ellas se encuentran todas aquellas relaciones con el sector industrial, el comercio, la salud, la educación, el transporte, el control de la infraestructura urbana y el medio ambiente. El software se ha convertido en un producto vital, tanto para empresas, organismos, servicios y tareas cotidianas de los ciudadanos como para la toma de decisiones, el intercambio de información y la gestión del conocimiento. (Fernández Pérez, 2018)

Esta situación conduce al incremento de la complejidad del software a partir de las funcionalidades que debe lograr. Surge con ello un ambiente de competencia y especialización entre las actividades productoras y comercializadoras que provoca un interés por lograr mayor calidad en los productos. Esto convierte a la calidad en un importante punto diferenciador entre las organizaciones a nivel mundial por las ventajas competitivas que puede aportar. (Fernández Pérez, 2018)

Las tendencias actuales consideran a la calidad como un factor estratégico. Romero y otros plantean: ‘‘(…) ya no se trata de una actividad inspectora sino preventiva: planificar, diseñar, fijar objetivos, educar e implementar un proceso de mejora continua, la gestión estratégica de la calidad hace de esta una fuente de ventajas competitivas que requiere del esfuerzo colectivo de

todas las áreas y miembros de la organización''. (Marin Diaz, Trujillo Casañola, & Buedo Hidalgo, 2019) (Jiménez Bibián, 2020) . La ingeniería de software desde sus inicios ha aplicado un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software, con el objetivo de alcanzar un software de alta calidad. Un software que sea producido en tiempo, con el costo planificado y que funcionen los requisitos pactados con el cliente. (Marin Diaz et al., 2019)

Las actividades de control y aseguramiento de calidad de los procesos y productos durante todo el ciclo de vida de software como parte de la Ingeniería de Software, permiten mayor utilidad con el propósito de ofrecer optimización, eficiencia y satisfacción de necesidades de los clientes. (Callejas-Cuervo, 2017) En Ingeniería de Software se han elaborado diferentes normas, modelos y estándares relacionados con las características del productos, las buenas prácticas, criterios entre otros aspectos que hay que tener en cuenta para que los productos de software cuentan con la calidad requerida. (NC, 2016) (Jiménez Bibián, 2020) La industria del software es uno de los sectores de más rápido crecimiento en el mercado por el incremento del papel de los productos de software en todas las esferas de la sociedad, sin embargo los resultados no se corresponden con la ejecución eficaz de los mismos. (Ahmed, Ahsan, & Abbas, 2016; Bannerman, 2013; Pressman, 2010) Sin embargo, son numerosos las investigaciones publicadas que reportan las dificultades que enfrentan los proyectos, por ejemplo el Standish Group en los reportes publicados en los últimos cinco años muestra que el comportamiento de los proyectos exitosos sigue siendo menor que los proyectos cancelados y fallidos. Los datos aportados reflejan que, a pesar de la creciente demanda de informatización en la sociedad, la producción de software todavía contempla problemas que inciden sobre el desarrollo de los productos. Una de las causas es la: Identificación de defectos luego de la implantación de los productos. (Johnson, 2016-2020)

Como industria, el software requiere de productos y servicios de alta calidad, lo cual se apoya con la aplicación de modelos, estándares y metodologías reconocidos internacionalmente (H. D. Ortiz Alzate, jun. 2016.) y una vía para el éxito de esta industria es certificar la calidad de procesos y producto. (Giraldo & Terán, 2019) Entre ellos se encuentran los estándares que definen un marco de trabajo para establecer el proceso de pruebas, como: ISO/IEC 14598: 2001 e ISO/IEC

25040:2011; otros, están dirigidos a evaluar características específicas de calidad o iniciativas regionales. (Fernández Pérez, 2018)

Para la evaluación y mejora de los procesos de software se pueden citar los estándares ISO/IEC 90003:2018, ISO/IEC 12207:2017, ISO/IEC 15504:2012. Se destaca también el Modelo de Capacidad y Madurez Integrado, en inglés Capability Maturity Model Integration (CMMI) creado por el Instituto de Ingeniería del Software (SEI) de la Carnegie Mellon University. Modelo que trabaja la mejora de procesos para el desarrollo de productos y servicios. Propone buenas prácticas que tratan las actividades de desarrollo y mantenimiento que cubren el ciclo de vida de los productos. (Fernández Pérez, 2018)

Estos modelos, normas y estándares explican qué prácticas deben institucionalizarse en las organizaciones, pero solo se alcanza los beneficios de estas cuando se complementan los procesos definidos, con los conocimientos y las habilidades del personal que las ejecuta. Las mejores prácticas se pueden definir como una serie de acciones, sistemas, herramientas y técnicas aplicadas y aprobadas con resultados sobresalientes en empresas que han sido reconocidas como de clase mundial según el Instituto Mexicano de Mejores Prácticas Corporativas. Para los efectos del control de la calidad del producto software existe el Comité Internacional de Cualificación de Pruebas de Software (International Software Testing Qualification Board, ISTQB por sus siglas en inglés), el cual está formado por más de 50 asociaciones nacionales de Control de Calidad del software y es el responsable de recopilar, documentar y certificar las mejores prácticas de la industria. Es una organización sin ánimo de lucro, fundada en el año 2002 en Escocia por un grupo de empresas, instituciones, organizaciones y personas especializadas en el campo de las pruebas y la industria del desarrollo de software. El fin de esta organización es brindar soporte y definir un esquema de certificación internacional. Dicho comité suministra un plan de estudios y un glosario de términos en los cuales se definen los estándares internacionales por niveles y se establecen las guías para la acreditación y evaluación de los profesionales en pruebas de software. Dichos planes incluyen y evalúan dentro del mismo los estándares, normas y modelos internacionales de control de calidad del software. (Fernández-Ocampo, 2017)

Según diversos autores en el área de calidad de software como Humphrey, Larman, Pressman, la calidad está altamente relacionada con los defectos en los productos y coinciden en que hay que invertir más en las actividades de control y aseguramiento de la calidad desde los inicios del desarrollo de software. (HUMPHREY, 1997.; Larman, 1999; Pressman, 2010) En correspondencia con los planteado por los autores antes mencionados, el ISTQB que estudia y da seguimiento al avance de las pruebas de software en más de 120 países, identificó en un reporte realizado que en el 2017 – 2018 las tendencias para la profesión de pruebas de software serán la automatización de pruebas, las pruebas ágiles y las pruebas de seguridad; y que, de las pruebas más importantes para la organización, se tiene las pruebas funcionales con un 83% y las pruebas de eficiencia del desempeño con un 60.7%. Sin embargo solo estos dos tipos de pruebas son ejecutadas en más del 50% de las organizaciones y 11 de los 17 tipos de pruebas solo son ejecutadas en menos del 20% de las organizaciones (Board(ISTQB), 2018), lo cual trae riesgos asociados a la satisfacción de los clientes y pérdida de mercado. (Chinarro Morales, 2019)

Pressman expone que: “Las pruebas del software son un elemento crítico para la garantía de calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación”. (R. Pressman, 2002) Las pruebas de software según Flores Mendoza son: “el proceso por el cual se encuentran bugs y errores en el software, las define como un conjunto de actividades sistemáticas y planeadas para descubrir errores en el software. Con frecuencia requieren mayor cantidad de esfuerzo realizado en un proyecto que cualquier otra actividad. Establece que las pruebas consumen al menos la mitad del tiempo y trabajo requerido para producir un programa funcional”. (Mendoza, 2019)

Los autores de este trabajo consideran que ambos conceptos tienen puntos en común, aunque no asocian las pruebas con la disminución de riesgos de fallos. Teniendo en cuenta las definiciones anteriores y la que plantea el ISTQB consideran para dicha investigación a las pruebas como: “una forma de evaluar la calidad y reducir los riesgos de fallos a partir de la comprobación del cumplimiento de las especificaciones del producto.” (Kramer & Legeard, 2016) Las pruebas para la investigación se consideran como actividades que representan una revisión final de las especificaciones, del diseño y de la codificación, con el objetivo de encontrar los posibles

fallos que puedan existir en el producto de trabajo a evaluar. Se propone que se realicen al producto de software, luego de culminada su implementación, y que se utilicen, además, para comprobar durante el desarrollo todas las unidades que se implementen. La frecuencia recomendada está sujeta a la culminación de los artefactos a evaluar". (Mendoz, 2019; Pressman, 2010)
Las pruebas son muy costosas por lo que se dejan para las últimas etapas del proyecto y no cubren todos los tipos de pruebas recomendadas. Por otra parte, las pruebas son el recurso más importante para la evaluación de la calidad de un software (Vásquez Romero, 2018) (Vera, Valdivia, Quentasi, Yana, & Apaza, 2020); sin embargo, calidad y pruebas no son lo mismo. La calidad se incorpora al software en todo el proceso de ingeniería, si no está ahí cuando comienza la prueba, no estará cuando se termine. Por esta razón deben enfocarse en la prevención y actividades de control desde el inicio del desarrollo del software. Tal y como plantea un principio de las pruebas. (David Flores Mendoza, 2019)

A continuación se listan los Principios de las pruebas (Kramer & Legeard, 2016)

1. Las pruebas demuestran la presencia de defectos, no su ausencia.
2. Las pruebas exhaustivas son imposibles.
3. Las pruebas tempranas ahorran tiempo, esfuerzo y costo.
4. Los defectos se agrupan.
5. Cuidado con la paradoja del pesticida.
6. Las pruebas dependen del contexto.
7. Falacia de la ausencia de errores.

La estrategia fundamentalmente trabajará sobre los principios de las pruebas tempranas y la dependencia de las del contexto asociado a los niveles de pruebas. Los autores de la investigación consideran que estos principios son importantes para la disminución de esfuerzo dedicado a las pruebas y la objetividad de las pruebas, aspectos que inciden en las entregas en tiempo y la satisfacción de los usuarios finales.

Como punto de partida para la realización de la estrategia se tendrán en cuenta los objetivos comunes de las pruebas establecidos por el ISTQB (Kramer & Legeard, 2016) para establecer los objetivos por niveles de prueba:

1. Evaluar los productos de trabajo tales como requisitos, historias de usuarios, diseño, código, entre otros.
2. Verificar si se han cumplido todos los requisitos especificados.
3. Validar si el objeto de prueba está completo y funciona como los usuarios y partes interesadas esperan.
4. Aumentar la confianza en el nivel de calidad del objeto de prueba.
5. Prevenir defectos.
6. Encontrar fallas y defectos
7. Proporcionar suficiente información para tomar decisiones en relación con el nivel de calidad del objeto de prueba.
8. Reducir el nivel de riesgo de una calidad de software inadecuada.
9. Cumplir con los requisitos y normas contractuales, legales o reglamentarias.

La realización de pruebas para evaluar el software se ha convertido en una tendencia en la industria, pero disimiles investigaciones y tendencias evidencian que se realizan luego de finalizado el producto y muchas veces solo se ejecutan pruebas funcionales. (Board(ISTQB), 2018) Esto supone un problema pues se aleja la detección del defecto del momento en que se introduce, lo que incide en los costos de corrección y alargan los cronogramas del proyecto, sin cubrir las pruebas estructurales y no funcionales. (HUMPHREY, 1997.) (Larman, 1999) (Pressman, 2010) (Raghuvanshi, 2020)

En la revisión bibliográfica realizada se pudo constatar que los autores consultados plantean la necesidad de las pruebas (Board(ISTQB), 2018; David Flores Mendoza, 2019; Fernández Pérez, 2018; Pressman, 2010) y de su correcto diseño, enmarcando las buenas prácticas que deben tenerse en cuenta para la realización exitosa de pruebas, pero no se hace una propuesta de qué probar y cómo probar. Por tanto, se establece como **problema** a resolver para esta investigación: ¿Qué y cómo probar

a lo largo del ciclo de vida del software teniendo en cuenta las buenas prácticas recomendadas en los modelos, normas y estándares más utilizados y las experiencias de investigadores?

El **objetivo** de la investigación consiste en desarrollar una estrategia de pruebas para evaluar el software que permita detectar los defectos más cercanos al momento en que se introducen y disminuir los riesgos de falla en operación.

Métodos o Metodología Computacional

Para el desarrollo de esta investigación se utilizaron los métodos que se mencionan a continuación. Además, se brinda una breve explicación de los fines para los que fueron utilizados.

Métodos teóricos:

1. Método dialéctico para el estudio crítico de los trabajos anteriores y para usar estos como fuente de referencia y comparación de los resultados.
2. El método analítico- sintético se utilizó para el estudio de la bibliografía acerca de los modelos de calidad más usados internacionalmente
3. El hipotético deductivo para la identificación de la situación problemática y de las soluciones.

Métodos empíricos:

1. Entrevista para obtener informaciones en pos de argumentar la situación problemática y la validación de los resultados.
2. La encuesta para obtener las experiencias de las organizaciones.
3. La observación participante para obtener la información necesaria para el planteamiento del problema, así como realizar la confrontación de los resultados obtenidos.
4. Métodos estadísticos para valorar el efecto de la propuesta.

5. El método experimental para comprobar la utilidad de los datos obtenidos a partir de la implementación de la estrategia general de pruebas.

Desde el punto de vista de una investigación una estrategia debe considerar las siguientes etapas según Ramírez y Lima: Diagnóstico, Planteamiento del objetivo general, Planeación estratégica, Instrumentación y Evaluación. (Cintra, Dihigo, & Hernández, 2020) Para esta investigación se realizó una alineación de estas etapas con las características propias de una estrategia de pruebas ("Model-Based Testing Essentials-Guide to the ISTQB Certified Model-Based Tester: Advanced Test Manager," 2016). Se identifica el objetivo general de la estrategia propuesta que estaría contenido en la política de calidad de la organización a partir de los resultados del diagnóstico. Se realiza la planeación estratégica definiendo los objetivos por niveles de pruebas y posteriormente se describe la instrumentación donde se explica qué probar a partir del tipo de prueba por niveles y el cómo probar a partir de describir los objetivos, objetos de pruebas típicos, bases de pruebas, enfoques y responsabilidades, defectos y tipos de fallas típicos y técnicas y estrategias. Por último, se muestran los resultados de la valoración de expertos y un estudio de casos para mejor comprensión de la propuesta.

Como primer elemento a tener en cuenta para el diseño y organización de las pruebas es definir la política de pruebas de la organización. (Chrissis, 2011; Kramer & Legeard, 2016) La misma debe describir los objetivos y metas de la organización para las pruebas. Constituye la filosofía de prueba, posiblemente en relación con las políticas de calidad de software de la organización. La política de pruebas debe abordar las actividades de prueba para nuevos desarrollos, así como para el mantenimiento. También puede hacer referencia a normas internas y/o externas para los productos de trabajo de pruebas y la terminología que se utilizarán en toda la organización.

Para la redacción de la política se propone la utilización de un grupo focal y/o tormenta de ideas donde participen miembros de la dirección de la organización que puedan tomar decisiones en

los procesos, los responsables de dirigir los procesos de calidad y algunos actores de estos procesos.

Luego de establecida la política de pruebas se diseña la **estrategia de pruebas** a ejecutar a partir de los niveles de prueba lo que hace la propuesta independiente del negocio, el tipo de producto y la metodología de pruebas. Las estrategias pueden ser ("Model-Based Testing Essentials-Guide to the ISTQB Certified Model-Based Tester: Advanced Test Manager," 2016) (Goericke, 2020):

1. Analíticas o basadas en datos: El Equipo de pruebas analiza la base de la prueba para identificar las condiciones a cubrir. Prueba basada en riesgos o requisitos. Basadas en riesgo y Basadas en requisitos
2. Basada en modelos: Elaboración de perfiles operativos. El equipo desarrolla un modelo del entorno.
3. Metódicas: Uso de listas de comprobación, por ejemplo, basado en ISO/IEC 25010: 2016.
4. De Conformidad con los procesos o estándares: Sujeto a normas de la industria o procesos como uno ágil
5. Reactivas o basadas en la experiencia: Como el uso de ataques basados en defectos, también pruebas exploratorias.
6. Consultivas: Pruebas dirigidas por el usuario
7. Con aversión a la regresión: Uso de Automatización extensiva
8. Combinadas (Combinación de las anteriores, es la más recomendada)

Las estrategias específicas seleccionadas deben ser apropiadas a las necesidades y los medios de la organización, y las organizaciones pueden adaptar las estrategias a operaciones y proyectos particulares. La estrategia de prueba puede describir los niveles de prueba que deben realizarse. En tales casos, debe proporcionar orientación sobre los criterios de entrada y salida para cada nivel y las relaciones entre los niveles.

Para determinar los tipos de estrategias a utilizar se puede utilizar el método de **grupo focal** o **tormenta de ideas** donde los participantes no deben estar definidos por sus cargos en la organización sino por sus habilidades y conocimientos. Se considera apropiado que deben participar especialistas que tengan en el desarrollo de software con énfasis en el análisis y las pruebas.

El diseño de la estrategia continua con la selección de los **niveles de pruebas** en los que se van a realizar las actividades. Los niveles de prueba se caracterizan con los siguientes elementos:

1. Objetivos específicos
2. Objeto de prueba
3. Base de prueba
4. Defectos y fallos típicos
5. Enfoques y responsabilidades

Según el ISTQB y los modelos, normas y estándares más utilizados a nivel internacional existen los siguientes cuatro niveles (Kramer & Legeard, 2016) (Goericke, 2020):

1. Nivel de aceptación: Las pruebas de aceptación se centran en el comportamiento y capacidades de todo el sistema o producto en el entorno real. En este nivel es necesario considerar los siguientes elementos:

    1. Se verifica la adecuación al uso del sistema por parte de usuarios de negocio.
    2. Las pruebas se realizan utilizando el entorno del cliente.
    3. El entorno de cliente puede reproducir nuevos fallos.
    4. Entre las formas comunes de pruebas de aceptación se encuentran: pruebas de aceptación de usuarios (UAT), pruebas de aceptación de operación (OAT), pruebas de aceptación contractual o reglamentaria, pruebas alfa y beta.

2. Nivel sistema: Las pruebas de **sistema** se centran en el comportamiento y capacidades de todo el sistema o producto. Este nivel tiene en cuenta los siguientes aspectos:

    1. La prueba tiene en cuenta el **punto de vista del usuario**.
    2. Implementación completa y correcta de los requisitos.
    3. Despliegue en el entorno de prueba simulando el entorno real con datos reales.
    4. Todas las interfaces externas son probadas emulando el entorno real del sistema.
    5. No se realizan pruebas en el entorno real.
    6. Los defectos inducidos podrían dañar el entorno real.

3. Nivel integración: Cada componente ya ha sido probado en la referente a su funcionalidad interna (prueba de componente). Las pruebas de integración comprueban las funciones externas tras las pruebas de componente. Comprueba la interacción entre los elementos de software (componentes) entre distintos sistemas o hardware y software (normalmente tras las pruebas de sistema).
4. Nivel componente (pruebas unitarias): Las pruebas unitarias son la primera fase de las pruebas que se le aplican a cada módulo de un software de manera independiente. Su objetivo es verificar que el módulo, entendido como una unidad funcional de un programa independiente, está correctamente codificado.

Los niveles de prueba que se van a proponer en la estrategia de prueba de cada organización pueden ser definidos por el mismo grupo y con los mismos métodos utilizados para definir los tipos de estrategias a aplicar. Los investigadores de este trabajo consideran que se debe evaluar el producto en todos los niveles de prueba, pues dejar de incluir alguno de ellos implicaría incurrir en riesgos que se deben mitigar en el próximo nivel. Aunque no es de carácter obligatorio, por ello se considera que si no se tienen en cuenta un nivel para su evaluación debe el grupo de trabajo redactar los riesgos con claridad para elaborar un plan de mitigación de riesgos en las siguientes etapas.

Para cada nivel de prueba es necesario especificar el objetivo, los objetos de pruebas típicos y la base de pruebas. Los objetivos definen el alcance de las pruebas, es decir hasta donde se va a probar. Los objetos de pruebas típicos son los productos de trabajo que se deben probar para cada nivel, estos pueden variar en dependencia de las necesidades y características del producto y su propio desarrollo. Las bases de pruebas son los elementos que van a constituir guías para evaluar los objetos de pruebas típicos.
Todos los **tipos de pruebas** se pueden realizar en cualquier nivel, depende de los objetivos de las mismas y la selección de técnicas a emplear.

Un tipo de prueba es un grupo de actividades de pruebas destinadas a probar características de calidad de un sistema de software, basada en objetivos de pruebas específicos. Los tipos de pruebas se dividen en 4 grupos ("Model-Based Testing Essentials-Guide to the ISTQB Certified Model-Based Tester: Advanced Test Manager," 2016) (Goericke, 2020):

1. Evaluar características funcionales (pruebas funcionales): se basan en funciones y características y su interoperabilidad con sistemas específicos, y puede llevarse a cabo en todos los niveles de pruebas.
2. Evaluar características no funcionales (pruebas no funcionales): Evalúa “cómo” funciona el sistema. Incluyen, pero no se limitan a, las pruebas de rendimiento, pruebas de carga, pruebas de estrés, pruebas de usabilidad, pruebas de mantenimiento, pruebas de fiabilidad y pruebas de portabilidad. Las pruebas no funcionales se pueden realizar en todos los niveles de pruebas.
3. Evaluar si la arquitectura es correcta y completa (pruebas estructurales): Puede realizarse en todos los niveles de pruebas. Las técnicas estructuradas son las mejores, usadas después de las técnicas basadas en especificaciones, con el fin de ayudar a medir el rigor de las pruebas mediante la evaluación de la cobertura de un tipo de estructura. La cobertura es la medida en que una estructura ha sido llevada a cabo por una serie de pruebas, expresado como porcentaje de los puntos que son cubiertos.

4. Evaluar los efectos del cambio (pruebas de Regresión y Confirmación (Re-prueba)): Las Re-Pruebas o Pruebas de Confirmación son aplicadas después que un defecto es identificado y corregido, con la finalidad de verificar que el defecto ya no se está presentando. Las Pruebas de Regresión se realizan sobre un componente ya probado, para verificar que no presenta nuevos defectos cuando se realiza una modificación después de dichas pruebas.

Para la propuesta de los tipos de pruebas a realizar por cada nivel se considera que se conforme un **grupo de trabajo técnico** con las habilidades anteriormente descritas (puede ser el mismo que se utilizó para la estrategia y los niveles) y que este proponga el conjunto de pruebas a realizar con las características a evaluar por cada nivel a un **grupo focal.** Este último debe aprobar la propuesta.

El tipo o la combinación de estrategias a aplicar, los niveles y los tipos de pruebas no deben ser camisas de fuerza. Para cada producto a probar debe existir un plan de pruebas donde se tome como punto de partida la estrategia de pruebas y se adapte según sea necesario por las características del producto a probar, así como la disponibilidad de tiempo de desarrollo. A partir de todos los elementos analizados, en el siguiente apartado se propone la estrategia de pruebas para organizaciones desarrolladoras de software.

Resultados y discusión

La propuesta se aplicará a la Universidad de las Ciencias Informáticas, en la cual se desarrollan productos de software de varios tipos de productos (portales e intranet, sistemas operativos, videojuegos, apk y sistemas de gestión (web y desktop)), para distintos negocios (gobierno, telemática, educativos, salud, deporte…) y se emplean para el desarrollo metodología basadas en historias de usuarios, casos de uso y descripciones de procesos.

La política de calidad que se identificó a partir del grupo focal fue:

Las actividades de Calidad se llevan a cabo para prevenir, controlar y mejorar la calidad de los productos. Las pruebas son actividades que representan revisiones de las especificaciones, del diseño, la codificación con el objetivo de encontrar los posibles fallos que puedan existir en el producto de trabajo a evaluar. Como estrategia se seleccionada combinar las estrategias analíticas: basada en requisitos y en riesgo, basadas en modelos, reactivas o basadas en la experiencia, metódicas y consultivas. Para la descripción del como probar se detallan para cada nivel de prueba: objetivos, objetos de pruebas típicos, bases de pruebas, enfoques y responsabilidades, defectos y tipos de fallas típicos y técnicas y estrategias. Ver tabla 1

Tabla 1 - Instrumentación de la estrategia de prueba.

	Componente	**Integración**	**Sistema**	**Aceptación**
Objetivos	1. Reducir el riesgo 2. Verificar los requisitos funcionales y no funcionales del componente 3. Encontrar defectos en el componente 4. Evitar que los defectos escapen a otros niveles de pruebas	1. Reducir el riesgo 2. Verificar los comportamientos funcionales y no funcionales de las interfaces 3. Crear confianza en la calidad de las interfaces 4. Encontrar defectos en la interacción entre componentes 5. Evitar que los defectos escapen a otros niveles de pruebas 6. Verificar las integración entre	1. Reducir el riesgo 2. Verificar que los comportamientos funcionales y no funcionales del sistema 3. Crear confianza en la calidad del sistema 4. Encontrar defectos 5. Evitar que los defectos escapen a producción	1. Validación de que el sistema está completo y funcionará según los previsto 2. Verificar que los comportamientos funcionales y no funcionales del sistema 3. Satisfacer requisitos de usuarios, operación y legales o reglamentarios

		sistemas		
Objetos de prueba típicos	1. Componentes/ clases/ unidades/ módulos 2. Código	1. Subsistemas 2. Bases de datos 3. Infraestructura 4. Interfaces	1. Aplicaciones 2. Configuración del sistema	1. Aplicaciones 2. Sistemas 3. Configuración del sistema y de datos
Base de prueba	1. Requisitos de componente 2. Diseño detallado 3. Código	1. Diseño de software y sistemas 2. Diagramas de secuencias 3. Especificaciones de interfaces y protocolos de comunicación 1. Casos de uso 2. Arquitectura	1. Especificación de requisitos de software y de sistema. 2. Informes de análisis de riesgos 3. Casos de uso, historias de usuario 4. Modelos de comportamiento del sistema 5. Manuales del sistema y del usuario	6. Procesos del negocio 7. Requisitos del usuario 8. Regulaciones, contratos legales. 9. Especificación de requisitos de software 10. Casos de uso, historias de usuario 11. Manuales del sistema, del usuario e instalación
Enfoques y responsabilidades	Es del desarrollador, se enfoca en la automatización de las pruebas, por lo general caja blanca. Estas pruebas deben ser llevadas a cabo por	La prueba de integración de componente es responsabilidad del desarrollador. Las pruebas de integración entre sistema es responsabilidad del	Probadores que tienen como tarea fundamental la detección de defectos a partir del diseño de la prueba. Se propone un equipo de pruebas	Son responsabilidad de los clientes, usuarios y operadores del sistema. Para este nivel si existe claridad sobre lo pactado entre las partes se propone que solo

	el equipo de proyecto.	probador. Estas pruebas deben ser llevadas a cabo por el equipo de proyecto.	independiente al equipo de desarrollo.	intervenga el equipo de desarrollo y el cliente.
Defectos y fallas típicos	Son en el entorno de los valores límites, funciones ausentes y validaciones.	Son datos incorrectos o faltantes, codificación incorrecta, secuencia incorrecta de llamadas, fallos o faltas de comunicación entre componentes. Estructura de mensajes inconsistentes e incumplimiento de normas de seguridad.	Problemas de validación, falta o fallo de las funcionalidades, falta de correpondencia con el manual de usuario.	Flujos de trabajo incorrectos, reglas de negocios implementadas incorrectoamente, fallos funcionales de seguridad, eficiencia del desempeño, portabilidad, compatibilidad.
Tipos de pruebas	1. Pruebas funcionales 2. No funcionales (seguridad) **3. Estructurales** 4. Pruebas de regresión y confirmación.	**1. Pruebas funcionales** 2. No funcionales **3. Estructurales** 4. Pruebas de regresión y confirmación.	1. Pruebas funcionales 2. No funcionales 3. Pruebas de regresión y confirmación.	1. Pruebas funcionales 2. No funcionales 3. Pruebas de regresión y confirmación.
Organización (Estrategias y pruebas)	1. Analítica (requisitos) 2. Metódica 3. Se recomienda en este nivel realizar una revisión completa tanto de los requisitos funcionales como de los no funcionales. 4. Para las pruebas estructurales el uso	1. Analítica (requisitos y riesgos) 2. Metódica 3. Basada en modelos 4. Para este nivel se recomienda una revisión de los requisitos funcionales y no funcionales con un enfoque analítico. En las pruebas estructurales se debe comprobar el	1. Analítica 2. Metódica 3. Basada en modelos 4. Reactivas o basadas en la experiencia. Se propone comenzar con una revisión completa de los requisitos funcionales y no funcionales. Teniendo en cuenta el tiempo de pruebas con	1. Analítica 2. Metódica 3. Consultivas En este nivel a diferencia del resto se recomienda comenzar con una metódica de las pruebas funcionales y no funcionales para validar que lo pacatado con el cliente está correctamente construido. Luego se

	de la metódica para comprobar el correcto uso de estándares de codificación. 5. Realizar las pruebas de confirmación para todos los defectos detectados en las pruebas anteriores y las de confirmación a partir del análisis de los riesgos. 6. En este nivel se propone la automatización de todas las pruebas propuestas.	cumplimiento de estandares y optimización de funciones, así como la correcta integración entre los componentes a través de modelos de arquitectura. Se le deben realizar pruebas de confirmación a todos los defectos detectados y la confirmación a parttir de un análisis de los riesgos.	el que se dispone se puede realizar a través de una analítica de requisitos o riesgos. Se debe comenzar con las pruebas funcionales con el fin de garantizar cierta estabilidad en el producto antes de comenzar con las pruebas no funcionales. Se pueden utilizar modelos realizados para la correcta conprensión del negocio del producto a través de flujos operacionales para las pruebas funcionales. Las pruebas de regresión y confirmación se deben utilizar para comprobar que todos los defectos detectados fueron correctamente corregidos y que no se introdujeron nuevos en dicho proceso.	puede continuar con un análisis de los riesgos que el cliente considere. Las pruebas de regresión y confirmación se proponen para realizarlas, pero es necesario tener claridad del alcance del contrato pactado pues pueden existir peticiones del cliente que no sean no conformidades sino peticiones de cambio. El enfoque consultivo se propone ya que en este nivel se considera que debe ser el cliente el que guíe las prueba.

Además de lo propuesto por cada uno de los niveles de pruebas, es necesario destacar que a juicio de los autores el nivel que se considera más crítico sin dejar de restarle importancia al resto es el **nivel sistema.** Constituye el último nivel donde los problemas en el producto dejan de considerarse defectos y pasan a ser no conformidades, por tanto, un factor fundamental es garantizar la independencia de las pruebas y la objetividad de las mismas. Como parte de la

estrategia de pruebas se incluyeron defectos y fallas típicos por cada nivel para ayudar en la detección de defectos.

Para la valoración de la propuesta en la solución del problema planteado, fue necesario realizar una encuesta para obtener los criterios de expertos. La selección de los expertos se hizo a través del análisis curricular. Participaron 17 expertos con más de diez años en la industria del software y de diferentes organizaciones desarrolladoras de software a nivel nacional.

Proceso de selección de expertos

Dada la variedad de instituciones que desarrollan software de la industria cubana y las características que esta posee, se hizo necesario tomar una muestra de expertos que tuviesen conocimientos amplios relacionados con la calidad y la gestión de la calidad en proyectos de software. Se realizó una valoración inicial de los posibles expertos para la validación del marco, considerando la experiencia práctica como el principal factor en esta investigación. Los criterios iniciales para la selección de expertos se listan a continuación:

1. Experiencia laboral en la industria de software de 10 años.
2. Producción científica enfocada al objeto a evaluar.
3. Haber desempeñado roles relacionados con la gestión de la calidad.

Al tener en cuenta estos criterios se realizó un cuestionario para el resumen curricular. Como resultado se seleccionaron 37 expertos a nivel nacional, de instituciones como: DESOFT, XETID, UCI y CUJAE. Incrementando los que han participado en la investigación en etapas anteriores con expertos internacionales con más de diez años en la industria del software y más de cinco años como consultores de la mejora de procesos de software.

Luego de seleccionados los expertos, teniendo en cuenta los conocimientos que poseen y el origen de los mismos y con el objetivo de validar la propuesta, se aplicó el método Delphi, el cual

tiene un amplio uso en varias áreas del conocimiento a nivel internacional y a nivel nacional, ha sido empleado fundamentalmente en investigaciones educativas y médicas, aunque en los últimos años se ha empleado en investigaciones de Ciencias Informáticas con el objetivo de socializar, externalizar y combinar el conocimiento de los expertos. El método aprovecha los elementos comunes en el grupo de expertos, preserva el anonimato mediante el uso de flujos de comunicación y permite la participación de expertos que se encuentren geográficamente dispersos (Trujillo, 2014).

A partir de los resultados arrojados se pudo contrastar que todas las categorías son evaluadas de Muy altas o Altas, validando la contribución del marco en la solución del problema de investigación. Para todas las categorías se obtuvo una moda[1] de Alta o Muy Alta. Los expertos no emitieron votos en la escala de Baja (2) o Ninguna (1). A partir de los votos emitidos por los expertos se obtienen los siguientes resultados:

Tabla 2 - Por ciento y moda a partir del criterio de los expertos.

Criterio	Porciento	Moda
Relevancia	94.7	5
Pertinencia	94	5
Coherencia	88.3	5
Comprensión	96.7	5
Exactitud	64.7	4

Elaboración propia

Teniendo en cuenta estos resultados se puede decir que los expertos coinciden en que el diseño de pruebas propuesto incorpora las buenas prácticas propuestas en los modelos, normas y estándares más utilizados. Adicionalmente se recibieron sugerencias por parte de los expertos:

1. Retroalimentar la estrategia a partir de su empleo en los proyectos.

2. Incorporar el diseño de pruebas por características de calidad del producto.

Conclusiones

1. La realización de pruebas de software es importante para reducir el riesgo de fallo en operación, sin embargo, muchas veces se realizan luego de finalizado el producto y solo se ejecutan pruebas funcionales.
2. La estrategia integra el qué probar a partir del tipo de prueba por niveles con el cómo probar a partir de describir los objetivos, objetos de pruebas típicos, bases de pruebas, enfoques y responsabilidades, defectos y tipos de fallas típicos y técnicas y estrategias.
3. Los resultados reafirman la necesidad de complementar el proceso de pruebas con una estrategia de prueba y se obtienen criterios positivos de los expertos.

Referencias

(NC), O. N. d. N. (2016). NC ISO/IEC 25010:2016 INGENIERÍA DE SOFTWARE Y SISTEMAS – REQUISITOS DE LA CALIDAD Y EVALUACIÓN DE SOFTWARE (SQuaRE) – MODELOS DE LA CALIDAD DE SOFTWARE Y SISTEMAS www.nc.cubaindustria.cu.

AHMED, M. A., AHSAN, I., & ABBAS, M. (2016). Systematic Literature Review: Ingenious Software Project Management while narrowing the impact aspect. Paper presented at the Proceedings of the International Conference on Research in Adaptive and Convergent Systems.

BANNERMAN, P. L. (2013). Barriers to project performance. Paper presented at the 2013 46th Hawaii International Conference on System Sciences.

BOARD (ISTQB), I. S. T. Q. (2018). Worldwide Software Testing Practices REPORT. Retrieved from

Callejas-Cuervo, M., A. C. Alarcón-Aldana and A. M. Álvarez-Carreño. . (2017). Modelos de calidad del software, un estado del arte. v 13 No. 1.

CINTRA, A. V., DIHIGO, A. G., & HERNÁNDEZ, Y. M. (2020). Strategy for developing non-functional requirements in health applications. Revista Cubana de Informática Médica, 12(1), 92-107.

CHINARRO MORALES, E. J. (2019). Definición e implementación del proceso de pruebas de software basado en la NTP-ISO/IEC 12207: 2016 aplicado a una empresa consultora de software.

DAVID FLORES MENDOZA, I. M., PAOLA TOVAR. (2019). Creación de una metodología óptima y eficiente para la implementación de pruebas de software. Instituto Politécnico Nacional de México. (c7.1726)

FERNÁNDEZ-OCAMPO, M. F. (2017). Propuesta de una metodología de control de calidad para los proyectos de automatización, basado en las mejores prácticas de ISTQB, caso: SOIN SA.

Fernández Pérez, Y. (2018). Modelo computacional para la evaluación y selección de productos de software.

GIRALDO, R. G., & TERÁN, L. M. J. (2019). Propuesta de laboratorio de certificación en la norma ISO 29119 de pruebas de calidad de software en el Centro de Servicios y Gestión Empresarial del SENA. Revista CINTEX, 24(2), 46-54.

GOERICKE, S. (2020). The Future of Software Quality Assurance: Springer Nature.

H. D. ORTIZ ALZATE, L. G. M. M., J. CARDEÑO ESPINOSA, Y N. C. ALZATE OSORNO, «», . (jun. 2016.). Impacto del uso de objetos interactivos de aprendizaje en la apropiación de conocimiento y su contribución en el desarrollo de competencias matemáticas: un resultado de experiencia de investigación. Rev. CINTEX, vol. 21, n.o 1, pp. 7188, vol. , n.o 1,, pp. 7188.

HUMPHREY, W. (1997.). Introducción al Proceso Software Personal.

JIMÉNEZ BIBIÁN, O. P. (2020). Pruebas de calidad aplicadas al sitio web Allison. Instituto Tecnológico de Colima.

JOHNSON, J. (2016-2020, 2020). The Standish Group International, Inc. Retrieved from standishgroup.com

KRAMER, A., & LEGEARD, B. (2016). Model-Based Testing Essentials-Guide to the ISTQB Certified Model-Based Tester: Foundation Level: John Wiley & Sons.

LARMAN, C. (1999). UML y Patrones. Introducción al análisis y diseño orientado a objetos. Prentice Halll.

MARIN DIAZ, A., TRUJILLO CASAÑOLA, Y., & BUEDO HIDALGO, D. (2019). Apuntes para gestionar actividades de calidad en proyectos de desarrollo de software para disminuir los costos de corrección de defectos. Ingeniare. Revista chilena de ingeniería, 27(2), 319-327.

MENDOZA , I. M., PAOLA TOVAR. (2019). Creación de una metodología óptima y eficiente para la implementación de pruebas de software. Instituto Politécnico Nacional de México. (c7.1726)

MODEL-BASED TESTING ESSENTIALS-GUIDE TO THE ISTQB CERTIFIED MODEL-BASED TESTER: Advanced Test Manager. (2016).

PRESSMAN. (2010). Software Engineering: A Practitioner's Approach, 7/e. RS Pressman & Associates. Inc, Vol. 73375977(McGraw-Hill).

PRESSMAN, R. (2002). Ingeniería de Software: Un enfoque práctico: Madrid: McGraw Hill.

RAGHUVANSHI, D. (2020). Introduction to Software Testing. International Journal of Trend in Scientific Research and Development (IJTSRD), Volume 4(Issue 3), 797-800.

TRUJILLO, Y. C. (2014). " MODELO PARA VALORAR LAS ORGANIZACIONES DESARROLLADORAS DE SOFTWARE AL INICIAR LA MEJORA DE PROCESOS". (Tesis de doctorado), Universidad de las Ciencias Informáticas.

VÁSQUEZ ROMERO, D. P. (2018). Estrategia de pruebas de software en un proyecto full stack para minimizar el impacto sobre los clientes y usuarios en una empresa de telecomunicaciones.

VERA, Y. P., VALDIVIA, J. J. G., QUENTASI, S. M. Z., YANA, D. M. C., & APAZA, R. E. C. (2020). Design Thinking en la Planificación de Pruebas de Software. Innovación y Software, 1(2), 40-51.

Conflicto de interés

Los autores de este artículo autorizan la distribución y uso de su artículo.

Contribuciones de los autores

Aymara Marin Diaz: La autora trabajó en el diseño de la investigación, en la estrategia de pruebas y en la validación de la propuesta.

Yaimí Trujillo Casañola: La autora trabajó en la propuesta del diseño de la investigación, los métodos científicos a utilizar y en las validaciones de la estrategia. Realizó revisiones del documento.

Denys Buedo Hidalgo: El autor trabajó en la estrategia de pruebas. Realizó revisiones a la investigación.