Revista Internacional de Investigación e Innovación Tecnológica

Página principal: www.riiit.com.mx

Modelo para la ejecución de pruebas de software

Model for the execution of software tests

Jiménez-Bibián, O.P.[a], García-Díaz, N.[a], Fajardo-Delgado, D.[b], Sánchez-Cervantes, M.G.[b], García-Virgen J.[a]

[a] División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Colima; C.P. 28976; Villa de Álvarez, Colima, México
{g1546010, ngarcia, jgarcia}@itcolima.edu.mx
[b] Departamento de Sistemas y Computación, Instituto Tecnológico de Ciudad Guzmán, C.P. 49100, Cd. Guzmán, Jalisco, México
{dfajardo, msanchez}@itcg.edu.mx

Innovación tecnológica: Clasificación de pruebas de software y una metodología para su aplicación.

Área de aplicación industrial: Ingeniería de Software y Productividad – Calidad.

Recibido: 19 Mayo 2017.
Aceptado: 25 Julio 2017.

Resumen

Las pruebas de software son un elemento crítico para garantizar la calidad de un producto de software. La necesidad de las compañías de desarrollar software de calidad, ha motivado el diseño de diversas metodologías para el desarrollo y ejecución de pruebas de software. Diversas compañías buscan implementar satisfactoriamente estas metodologías, pero definir los pasos para mejorar y controlar las fases del proceso de pruebas de software y el orden en que estas se implementan es, en general, una tarea difícil. En el presente artículo se propone una nueva metodología para el desarrollo y ejecución de pruebas de software. Dicha propuesta consiste de una versión híbrida de los modelos TMMi (Testing Maturity Model integrated), MoProSoft (Modelo de Procesos de Software) y del Proceso de Pruebas de Software para Ambientes Universitarios. A diferencia de estos modelos, la metodología propuesta es adecuada para su aplicación en desarrollos de software a pequeña y mediana escala, omitiendo detalles que dificultan el seguimiento de la ejecución de las pruebas y permiento adaptar el modelo para diferentes plataformas de cómputo (Web, móviles y de escritorio). Para mostrar la efectividad de la metodología propuesta, se llevó a cabo un caso de estudio para el sitio Web Allison. Este sitio Web es un producto desarrollado por la empresa mexicana Maldonado Software. Como parte de la investigación en el caso de estudio, y como una contribución adicional, se propuso una nueva clasificación de las pruebas de software de acuerdo a su naturaleza de ejecución, definiendo cuales de ellas son de caja negra y/o caja blanca. En la aplicación de la metodologia propuesta, se

realizaron un total de 51 pruebas, de entre las cuales se lograron identificar oportunamente 28 errores de diferente gravedad. Los resultados muestran la efectividad de la metodología al integrar las mejores prácticas de los modelos robustos aplicables al desarrollo de software a pequeña y/o mediana escala.

Palabras clave: Modelos de pruebas de Software, calidad de software, herramientas de prueba.

Abstract

Software testing is a critical element to guarantee the quality of software. The need of companies to develop software with quality has motivated the study of several methods for the development and execution of software testing. Implementing these methods requires defining the process to improve and control the phases of the software testing and their execution order, which is a hard task. In this paper, we propose a new method for the development and implementation of software testing. This method consists of the combination of the models TMMi (testing maturity model integrated), MoProSoft (software process model) and the software testing process for the university environment. In comparison with these models, our proposed method is adaptable for software developments for small to medium scale projects. It also adapts to the software development for different platforms, such as Web, mobile, and desktop. We show the effectivity of our proposed method through a study case for the website Allison. This website is a software developed by the mexican company Maldonado Sofware. As an additional contribution, we proposed a new classification of software testing according to its type of black box or white box testing. We performed 51 software testing from which, we identified 28 different types of failures. The results of the study case show the effectivity of our proposed method.

Key words: Software testing, software quality, testing tools.

1. Introducción

Las pruebas de software es una de las actividades más importantes en el desarrollo de un software. Generalmente, estas consisten de una revisión extensa del cumplimiento y validez de las especificaciones, diseño y codificación de un producto final de software. De acuerdo a Amo *et al.* (2005), las pruebas de software son un elemento crítico para la garantía de 'calidad' del software.

La necesidad de garantizar software de calidad, ha motivado el diseño de diversas metodologías para el desarrollo y ejecución de pruebas de software. Sin embargo, muchas compañías presentan dificultades en adaptar una metodología de pruebas durante el desarrollo del software (Camargo *et al.*, 2013). Lo anterior debido a que, en la práctica, definir los pasos para mejorar y controlar las fases de un proceso de pruebas de software y el orden que estas se implementan es, en general, una tarea difícil (Andersin, 2004; Camargo *et al.*, 2013).

Existe en la literatura diversas investigaciones referentes a metodologías para el desarrollo y ejecución de pruebas de software, de entre las que destacan (Burnstein, 1996; Koomen & Pol, 1999; ISO/IEC, 2013; Guardati & Ponce, 2011; Jústiz-Núñez *et al.*, 2013). En el ámbito internacional, los modelos TMMi (Testing Maturity Model integrated) (Burnstein, 1996), TPI (Test Process Improvement) (Koomen & Pol, 1999) y el estándar ISO/IEC 29119-2 (2013), contribuyen a la

práctica disciplinada de pruebas de software (García *et al.*, 2014). Estos modelos actualmente representan los estándares de la industria del desarrollo de software, y de ellos se derivan otras metodologías más; en (von Wangenheim, 2010) se encuentra una extensa revisión de estos modelos y sus derivados. Sin embargo, estos modelos son demasiado amplios y robustos, de tal modo que se dificulta su aplicación en proyectos de desarrollo de software a pequeña y mediana escala.

Para proyectos más modestos de desarrollo de software, destaca la metodología de pruebas de software de Jústiz-Núñez *et al.* (2013). Esta metodología es asequible en comparación con las mecionadas anteriormente, y consta de cuatro niveles y de un conjunto de roles que facilitan la ejecución de pruebas. Sin embargo, aunque esta metodología contempla la ejecución de pruebas en las etapas iniciales del desarrollo de software, carece de un nivel para las actualizaciones y mantenimiento del mismo que surgen en ambientes no controlados de pruebas.

Por otra parte, en México, Guardati & Ponce (2011) presentan una Guía de Pruebas de Software que complementa al Modelo de Procesos para la Industria de software (MoProSoft). Esta guía provee plantillas, ejemplos y valores estándares de indicadores de prueba que pueden utilizarse como base para definir los documentos propios en cada organización. Sin embargo, para llevar a cabo las pruebas se requiere que el proceso de desarrollo de software se base en MoProSoft, lo que dificulta su individualización a la hora de aplicación en los proyectos de desarrollo de software. (Algo similar ocurre con TMMi que complementa a modelo CMMi (Capability Maturity Model integration) durante las fases de pruebas).

En el presente trabajo, se propone una nueva metodología para el desarrollo y ejecución de pruebas de software. Esta metodología se inspira en las metodologías de Burnstein (1996), Guardati & Ponce (2011), y Jústiz-Núñez *et al.* (2013), donde el resultado es una versión híbrida para aplicación de pruebas de software en pequeña y mediana escala.

El resto del documento se organiza de la siguiente manera. La sección 2 presenta una breve descripción de las pruebas de software y se propone una clasificación de cuáles pertenecen a pruebas de tipo caja blanca y/o caja negra. La sección 3 explica las metologías seleccionadas para crear la metodología híbrida. La sección 4 presenta la metodología propuesta para la aplicación de las pruebas de software. La sección 5 muestra el caso de estudio en el que se aplicó la metodología híbrida para la ejecución de pruebas de software. La sección 6 presenta los resultados obtenidos de la ejecución de las pruebas de software. Finalmente, la sección 7 presenta las conclusiones de la investigación.

2. Pruebas de software y su clasificación

Las pruebas de software son un elemento crucial para evaluar la calidad de un producto de software. De acuerdo a Pan (1999), las pruebas de software se pueden clasificar con base en su propósito, por la fase del ciclo de vida y por su alcance. Respecto al próposito de las pruebas de software, estas pueden clasificarse como: pruebas de corrección, pruebas de rendimiento, pruebas de fiabilidad y pruebas de seguridad. Con base al ciclo de vida, las pruebas de software pueden clasificarse como: pruebas de fase de requisitos, pruebas de fase de diseño, pruebas de fase de programa, evaluación de resultados de pruebas, pruebas de fase de instalación, pruebas de aceptación y pruebas de mantenimiento. Finalmente, las pruebas de software pueden clasificarse por el alcance que tienen como: pruebas unitarias, pruebas

de componentes, pruebas de integración y pruebas de sistemas. Por otra parte, Estrada *et al.* (2011) clasifica las pruebas de software acorde a las características de calidad de la norma ISO 9126, donde define criterios de funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

Un criterio adicional de clasificación para las pruebas de software es la disponibilidad de acceso al código fuente. De esta forma, las pruebas pueden clasificarse del tipo 'caja blanca' o 'caja negra'. Las pruebas de tipo *caja blanca* se basan principalmente en la estructura interna del software y son muy efectivas en la validación del diseño y en la búsqueda de errores de programación y de implementación. Las pruebas de *caja negra* se enfocan en los requerimientos funcionales del software y se llevan a cabo desde la perspectiva del usuario final. Sin embargo, hasta donde los autores del presente trabajo conocen, no existe en la literatura una clasificación de las principales pruebas de software para saber si pertenecen a las de tipo de caja blanca o caja negra. En la Tabla 1, se muestra una clasificación de este tipo.

Tabla 1. Clasificación de las Pruebas de Software.

Prueba de Software	Descripción	Caja Negra	Caja Blanca
Pruebas de Unidad	Generalmente se realizan con acceso al código fuente y en etapas tempranas de desarrollo.	-	X
Pruebas de Integración	Integran los módulos que componen al sistema y corrigen lo errores asociados a la interfaz. Para ello, comúnmente requieren del acceso al código fuente.	-	X
Pruebas del Sistema	Evalúan los requisitos no funcionales del sistema, tales como seguridad, rendimiento, confiabilidad, interconexiones externas, dispositivos, entre otros.	X	X
Pruebas de Validación	Se enfocan en las acciones visibles para el usuario y las salidas del sistema, y se realizan como si se tratara del usuario final.	X	-
Pruebas de Recuperación	Evalúan la habilidad del sistema para recuperarse a fallos.	X	X
Pruebas de Rendimiento	Consisten en verificar el rendimiento de un módulo del sistema o del sistema en sí durante su ejecución.	X	X
Pruebas de Seguridad	Verifican que los mecanismos de protección de un sistema realmente protegen de intrusiones externas o acciones malintencionadas.	X	X
Pruebas de Despliegue	Miden el desempeño del producto de software sobre diferentes plataformas y entornos de operación.	X	-
Pruebas de Resistencia	Confrontan el software ante situaciones anormales de sobrecarga de datos, memoria y número de interrupciones.	X	X
Pruebas de Picos de Carga	Inicia considerando un volúmen normal de usuarios concurrentes en el software para posteriormente introducir picos de carga. Al finalizar el pico de carga, el volúmen de usuarios vuelve a la normalidad.	X	X
Pruebas de Estrés	Determinan el punto de ruptura del sistema incrementando linealmente el número de usuarios.	X	X
Pruebas de Carga	Evalúan el comportamiento del sistema bajo una cantidad de peticiones por segundo. Generalmente se realiza durante las últimas fases del desarrollo.	X	-
Pruebas de Aceptación	Comparan el producto final del software con los requisitos del cliente desde la perspectiva del usuario final.	X	-
Pruebas de Instalación	Evalúan el comportamiento del software durante su instalación en un entorno final.	X	-
Pruebas de Regresión	Verifican que los cambios realizados al software no producen efectos negativos.	X	X
Pruebas de Facilidad de Uso	Evalúan que tan fácil es para al usuario final aprender a utilizar el software.	X	-
Pruebas de Interfaz Gráfica de Usuario	Evalúan los componentes de la interfaz y la interacción del usuario final con el sistema.	X	-

Pruebas Alfa	Utilizan el software en un ambiente no controlado por el usuario final y con la presencia del desarrollador.	X	-
Pruebas Beta	Se ejecutan en un ambiente no controlado por el desarrollador y en un sitio seleccionado por el usuario final.	X	-

3. Preliminares

En esta sección se describen las metodologías que sirvieron como base para el desarrollo de la metodología híbrida descrita en la sección 4. Las metodologías de referencia son: el TMMi (2017), MoProSoft (2017) y el Proceso de Pruebas de Software para Ambientes Universitarios de Jústiz-Núñez *et al.* (2014).

El TMMi (Test Maturity Model integration) es un modelo de referencia que utiliza conceptos de niveles de madurez para la evaluación y mejora de procesos (Sanz *et al.*, 2008). La aplicación del TMMi permite un proceso de pruebas controlado y estructurado, una calidad del producto de alto nivel, y comúnmente, tiempos cortos de seguimiento (Veenendaal & Wells, 2012). La Figura 1 muestra los niveles de madurez que conforman al modelo TMMi. Este modelo se considera un estándar de la industria y está muy relacionado con el CMMi (Capability Maturity Model integration), al grado de visualizarse como un complemento de este último, respectivamente. Debido a la trayectoria y su relación con CMMi, se considera que TMMi posee una robustez a diferencia de otros modelos. Sin embargo, TMMi contempla un gran número de prácticas que deben cumplirse, aunque no todas ellas sean factibles para todos los tamaños de equipos de desarrollo ni compañías (Camargo *et al.*, 2013).

Figura 1. Niveles de madurez y prevención de defectos (TMMi, 2017).

MoProsoft es un modelo de procesos que incorpora aspectos de negocios y buenas prácticas de la ingeniería de software, ajustadas para las pequeñas y medianas empresas de la industria del Software en México (Vega, 2014). Guardati y Ponce (2011) desarrollaron una guía que complementa a MoProSoft en el área de pruebas, definiendo para cada una de las categorías del modelo las actividades de pruebas necesarias para garantizar la calidad del producto. Además, incluyeron algunas actividades de pruebas no contempladas en el modelo, debido a que son de fundamental importancia para aumentar la competitividad y permanencia en el mercado de este tipo de empresas. Para facilitar la aplicación de las pruebas, la Guía de Pruebas de Software provee plantillas y ejemplos que pueden ser usados como base para definir los documentos propios en cada organización. La Figura 2 muestra el modelo de procesos de MoProSoft.

Figura 2. Modelo de procesos de sofwtare (MoProSoft, 2005).

Por otro lado, Jústiz-Núñez *et al.* (2014) presentan una metodología de pruebas del software para el laboratorio de calidad del Instituto Superior Politécnico "José Antonio Echeverría" en Cuba. Ellos definen una serie de niveles y un conjunto de roles que estructuran la planeación y ejecución de las pruebas de software. La Figura 3 muesta los niveles de prueba de la metodología de Jústiz-Núñez *et al.*, (2014).

Figura 3. Niveles de pruebas de la metodología de Jústiz-Núñez *et al.*, (2014).

4. Metodología híbrida

En el presente trabajo, se propone una nueva metodología para la planeación y ejecución de pruebas de software. Dicha metodología es una versión híbrida de los modelos TMMi (2017), MoProSoft (2017) y del Proceso de Pruebas de Software para Ambientes Universitarios de Jústiz-Núñez *et al.* (2014). La metodología propuesta consiste de una estructura de seis niveles de pruebas secuenciales y de un conjunto de roles para la ejecución de las pruebas de software (ver Figura 4). Dicha estructura de niveles, no permite ejecutar un nivel *i* de la metodología sin antes haber concluido satisfactoriamente el nivel *i*-1.

A continuación, se describen los niveles de prueba de la metodología propuesta.

4.1 Nivel 1 - Inicial

Este nivel consiste en recolectar toda la información que sea posible para la ejecución de las pruebas de software. Incluye la información de la organización, empleados, departamentos, proyecto de software hasta la indagación de las diferentes pruebas que existen. Esto con el objetivo de enriquecer a los encargados de pruebas el funcionamiento de la empresa, así también como del software que se desarrollará. El líder de pruebas deberá involucrarse en todos los procesos de la empresa posible, así también durante todas las fases del ciclo de vida de software para la comprensión del software a desarrollar.

4.2 Nivel 2 - Selección

Durante este nivel se seleccionan las pruebas que se aplicarán al producto de software. El líder de pruebas determina cuáles de ellas son las de mayor adaptabilidad y efectividad para cumplir con los objetivos del proyecto a desarrollar. Para realizar la selección de las pruebas de software, el líder previamente a este nivel debió haber realizado un estudio detallado de la organización de la empresa, levantamiento de requerimientos y análisis del sistema en general.

4.3 Nivel 3 - Planificación

Este nivel se basa en el nivel 'Inicial' de la metodología de Jústiz-Núñez *et al.* (2014), y consiste en realizar una planeación estratégica y calendarizada de las pruebas en

todo el ciclo de vida del software. Para ello se contempla la elaboración de un 'plan de pruebas de software' con base en el formato de la guía de pruebas de Guardati & Ponce (2011).

4.4 Nivel 4 - Ejecución de Pruebas

Este nivel se encarga de la ejecución y documentación de todas las pruebas seleccionadas durante el nivel 2 para el producto de software. Lo anterior conforme al plan de pruebas y a la calendarización definida en el nivel 3. Esta es una de las dos fases iterativas de la metodología híbrida, por lo que se puede repetir tantas veces sea necesario hasta lograr su cometido.

4.5 Nivel 5: Liberación

Este nivel toma como referencia el nivel de 'Optimización' del TMMi, el cual consiste prevenir los defectos del software y permitir la implementación de controles de calidad. Para ello, en este nivel se corrigen la mayor cantidad de errores del nivel 4 para que, posteriormente, el líder de pruebas decida si el software es apto para liberarse y proceder con la instalación o si es necesario regresar al nivel 4 para someter nuevamente el software a una nueva fase de pruebas.

4.6 Nivel 6: Mantenimiento

Este nivel es el último de la metodología propuesta, y se basa en el proceso de 'Mantenimiento' de MoProSoft para llevar a cabo las pruebas de las actualizaciones del software. Este es el segundo nivel iterativo de la metodología híbrida, consiste probar cada actualización del sistema, garantizando que las nuevas modificaciones no repercutan en la estabilidad del producto de software.

Figura 4. Modelo Híbrido.

La Tabla 2 muestra los roles que intervienen para el desarrollo de las pruebas de software y las responsabilidades de cada rol.

Tabla 2. Roles y actividades.

Rol	Responsabilidad
Líder de pruebas	El líder de pruebas es el encargado de diseñar el plan de pruebas de software, así como de asegurarse de que se cumpla todo lo estipulado en dicho plan. También, administra y monitorea el avance de cada una de las pruebas programadas, y aprueba la validación del software.
Encargado de pruebas	El encargado de pruebas tiene como objetivo ejecutar las pruebas asignadas por el líder de pruebas y el cumplir con los requisitos del plan de pruebas en tiempo y forma.
Encargado auxiliar de pruebas	Es una persona externa a la empresa o el proyecto, cuyo objetivo es evaluar el software como producto final y retroalimentar al proyecto de desarrollo antes de la liberación del software.

5. Caso de estudio

La empresa Maldonado Software (MSW, 2017), ubicada en el estado de Colima, México, desarrolló el sitio Web Allison, una página Web destinada a la búsqueda de inmuebles. La Figura 5 muestra una captura de pantalla del sitio Web Allison.

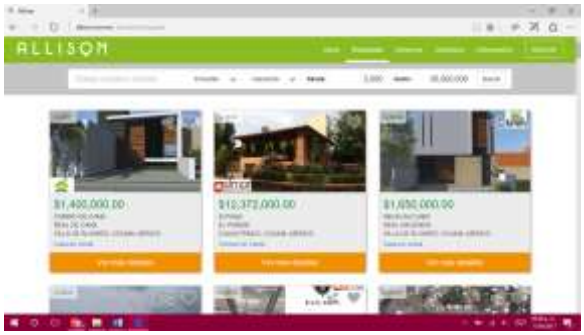

Figura 5. Ejemplo de pantalla de sitio Web Allison.

La empresa requería de la realización de pruebas de software de caja negra para garantizar la calidad del sitio Web Allison. Para cumplir con el objetivo, se implementó la metodología propuesta en este trabajo, considerando únicamente pruebas de software del tipo de caja negra. Esto debido a la renuencia de la empresa en conceder el acceso al código fuente. La Tabla 3 muestra las pruebas de software de tipo caja negra que se emplearon en el caso de estudio.

A continuación, se describen cómo se implementaron los niveles de prueba de la metodología propuesta.

5.1 Nivel 1

Este nivel consistió principalmente en realizar una entrevista con el personal de Maldonado Software (2016). Esto con el objetivo de conocer la forma como trabaja la organización y acerca de los requerimientos y especificaciones del proyecto. También en este nivel, se enriqueció el conocimiento a través de (Pressman, 2015; Medina (2014); y SWEBOK, 2004) sobre la diversidad de pruebas de software que pudieran ser aplicables en el proyecto.

5.2 Nivel 2

En este nivel se seleccionaron las pruebas de software del tipo de caja negra más adecuadas para el proyecto. En este caso, las pruebas de la Tabla 3 fueron las elegidas para el sitio Web Allison.

Tabla 3. Pruebas de software seleccionadas para el sitio Web Allison.

Prueba de Software	Implementación en el proyecto
Prueba de Estrés	Esta prueba evalúa el número de usuarios que ingresan al sitio Web Allison por medio de un software especializado. Este software parte de un conjunto de datos de entrada relativamente pequeño, y que posteriormente incrementa cada cierto periodo de tiempo. Después de alcanzar un número de datos de entrada relativamente alto, se prueba la funcionalidad del servidor y/o aplicación.
Prueba de Validación	La ejecución de esta prueba consistirá en validar cada uno de los campos del sistema para intentar encontrar irregularidades en el mismo, así también como validar que los campos numéricos no acepten caracteres alfabéticos por mencionar un ejemplo.
Prueba de Seguridad	Al ser un sistema Web, se requiere de la comprobación de cada uno de los hipervinculos para validar la lo siguiente: 1) que el usuario no puede entrar a un módulo sin antes autenticarse; 2) que el usuario no pueda acceder a la información de otro usuario colocando únicamente o cambiando el hipervínculo; y 3) validar que los hipervínculos no estén "rotos".
Prueba de Despliegue	Esta prueba consiste en la visualización del sistema en diferentes navegadores como: Microsoft Edge, Mozilla Firefox, Google Chrome, Safari, Vivaldi y Opera. Así como también diversos Sistemas Operativo como: MacOS, iOS, plataformas Windows, distribuciones Linux y Android.
Pruebas de Interfaz Gráfica de Usuario	El objetivo principal de esta prueba es asegurar que la interfaz gráfica de usuario tiene una apropiada navegación a través de los diferentes módulos del sistema.

5.3 Nivel 3

Durante el nivel 3, se desarrolló un plan de pruebas de software con base en la Guía de Pruebas de Guardati & Ponce (2011). En

dicho plan, se redacta la información de la empresa, objetivos del proyecto y de cada una de las pruebas, formatos para la ejecución de las mismas, calendarización, hardware y software requerido, entre otros.

5.4 Nivel 4

Durante esta fase se aplicaron diversas pruebas de software entre las cuales se pueden mencionar: pruebas de estrés, validación, seguridad, despliegue y de Interfaz Gráfica de Usuario, tal y como se menciona en la Tabla 3. Todas estas pruebas se aplicaron tal cual marca el plan de pruebas de software con los formatos propuestos.

5.5 Nivel 5

Durante este nivel se realizó un informe que describe las pruebas de software que se realizaron y cuáles fueron los resultados obtenidos. Este informe se entregó al jefe de desarrollo del proyecto para la corrección de los errores identificados.

5.6 Nivel 6

Este nivel es el único que no se aplicó al desarrollo de Allison (2017), debido a que aún no se han contemplado actualizaciones al Sitio Web.

En el presente artículo se muestran los resultados de la aplicación de la metodología híbrida solo en el caso de estudio del proyecto de Maldonado Software, debido a que la empresa se encontraba en esos momentos desarrollando el sitio web Allison y no tenía otros proyectos en desarrollo donde se pudiera volver a probar esta metodología. Sin embargo, esto no descarta que la metodología propuesta sea viable a diferentes proyectos de la industria del software, principalmente los de pequeña y mediana escala a los cuales esta enfocada.

6. Resultados

Se realizaron un total de 51 pruebas de software para el sitio Web Allison, de las cuales 23 no mostraron errores. Las otras 28 mostraron errores de diferente magnitud. A pesar de los errores encontrados el sistema es usable por un usuario final con conocimientos básicos en informática. En lo que corresponde a los errores es posible corregirlos en un corto periodo de tiempo sin afectar el sitio Web Allison ni a los usuarios. La Figura 6 muestra un gráfico del total de pruebas aplicadas.

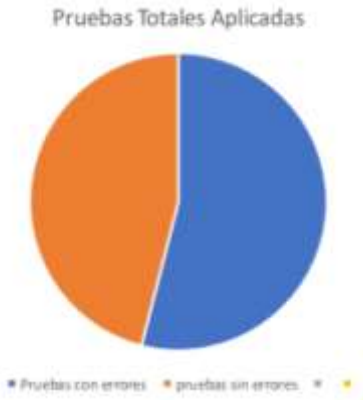

Figura 6. Total de pruebas aplicadas.

La Tabla 4 muestra los resultados totales separados por cada tipo de prueba aplicada en el sitio Web de Allison.

Tabla 4. Total de pruebas aplicadas separadas por categoría.

Tipo de prueba	Total de pruebas aplicadas	Total de pruebas con errores	Total de pruebas sin errores
Pruebas de Estrés	4	0	4
Pruebas de Despliegue	14	14	0
Pruebas de Validación	19	2	17
Pruebas de	3	2	1

Seguridad			
Pruebas de GUI	11	10	1

7. Conclusiones

En este trabajo, se propone una nueva metodología para el desarrollo y ejecución de pruebas de software. Esta metodología permite capturar los beneficios de los modelos robustos de TMMi, MoProSoft y del Proceso de Pruebas de Software de Jústiz-Núñez *et al*. (2014), para su aplicación en proyectos de desarrollo de software de pequeña y mediana escala.

Los resultados obtenidos en esta investigación por parte de las pruebas de software nos permitieron conocer el estado de calidad del Sitio Web Allison. Los resultados reflejan que en más de la mitad de las pruebas aplicadas presentaron distintos tipos de errores, principalmente de inconsistencia de las interfaces. Este tipo de errores puede dificultar la navegación por el sistema, pero aun así es usable por el usuario final. En lo que concierne a la validación y seguridad del software, el sistema no presentó ningún error que pudiera comprometer la información. Por tal motivo se puede concluir que la metodología fue implementada con éxito ya que permitió encontrar una gran cantidad de errores y conocer el estado de calidad en el que se encuentra.

Sin embargo, como toda metodología, esta tiene que ir evolucionando para poder adaptarse a los diferentes panoramas que se encuentran en la industria. Por lo que es probable que se hagan futuras adecuaciones o incluso se pueda expandir la metodología propuesta.

Adicionalmente, en este trabajo también se propone una clasificación de las pruebas de software para determinar si son de tipo caja blanca y/o caja negra. Esta clasificación permite determinar la prueba más adecuada con base al acceso al código fuente del software que se desea probar.

Finalmente, como trabajo futuro, se propone implementar la aplicación de la metodología desarrollado en este artículo, a otros casos de estudio para evaluar su efectividad. Además, se tiene planeado robustecer la metodología con instrumentos como lo son las métricas de pruebas de software (Número de casos de prueba pasados, fallidos, bloqueados, número de defectos encontrados, número de defectos aceptados, etc.), un plan de implementación en entornos de usuario y un plan de pruebas estándar para la metodología propuesta.

Agradecimientos

Los autores del presente trabajo agradecen al Consejo Nacional de Ciencia y Tecnología (CONACYT) y al Programa para el Desarrollo Profesional Docente (PRODEP), por el apoyo brindado el apoyo durante el desarrollo de esta investigación. También, se agradece a los Institutos Tecnológicos de Colima y de Ciudad Guzmán, por las facilidades prestadas para la colaboración entre investigadores de ambas instituciones. Finalmente, se agradece a la empresa Maldonado Software, por las facilidades para la utilización del software y el análisis del caso de estudio.

Referencias

1. **Andersin, J. (2004).** TPI–a model for Test Process Improvement. Department of Computer Science, University of Helsinki (http://www. cs. helsinki. fi/u/paakki/Andersin. pdf), Helsinki.

2. **Burnstein, I., Suwanassart, T., & Carlson, R. (1996, October).**

Developing a testing maturity model for software test process evaluation and improvement. In Test Conference, 1996. Proceedings., International (pp. 581-589). IEEE.

3. **Camargo, K. G., Ferrari, F. C., & Fabbri, S. C. P. F. (2013, October).** Identifying a subset of TMMi practices to establish a streamlined software testing process. In Software Engineering (SBES), 2013 27th Brazilian Symposium on (pp. 137-146). IEEE.

4. **Estrada, A. F., García, T. C., Perdomo, Y. L., Cintra, A. V., Martínez, R. D., & Díaz, R. C. (2011).** Una experiencia novedosa para el testing desarrollada por un departamento de pruebas de software. Revista Cubana de Ciencias Informáticas, 5(2).

5. **Fernando Alonso Amo, Francisco Javier Segovia y Loïc Martínez Normand.** Introducción a la Ingeniería de Software: Modelos de Desarrollo de Programas. Editores, Javier Barbero Rubio y Luis Egüen. Madrid, España en Delta Publicaciones. Primera Edición, 2005. ISBN 84-96477-00-2.

6. **Garcia, C., Dávila, A., & Pessoa, M. (2014, November).** Test process models: Systematic literature review. In International Conference on Software Process Improvement and Capability Determination (pp. 84-93). Springer International Publishing.

7. **Guardati, S., & Ponce, A. (2010).** Guía de Pruebas de Software (GPS) para MoProSoft.

8. **IEEE Computer Society, ACM**. Software Engineering Body of Knowledge (SWEBOK). SWEBOK executive editors, Alain Abran, James W. Moore; editors, Pierre Bourque, Robert Dupuis. (2004). Pierre Bourque and Robert Dupuis. ed. Guide to the Software Engineering Body of Knowledge - 2004 Version. IEEE Computer Society. pp. 1–1. ISBN 0-7695-2330-7.

9. **ISO/IEC: ISO/IEC/IEEE 29119-2:2013** Software and systems engineering – Software testing – Part 2: Test processes. Geneva (2013).

10. **Jústiz-Núñez, Dalila, Gómez-Suárez, Darlene, & Delgado-Dapena, Marta Dunia**. **2014.** Proceso de pruebas para productos de software en un laboratorio de calidad. Ingeniería Industrial. 35(2), 131-145.

11. **Koomen, T., & Pol, M. (1999).** Test process improvement: a practical step-by-step guide to structured testing. Addison-Wesley Longman Publishing Co., Inc.

12. **Medina Javier. (2014).** Pruebas de Rendimiento TIC. SG6 C.B. P. 34. ISBN: 978-1-312-31567-9.

13. **"MoProSoft". 15 de Abril de 2017,** de https://www.researchgate.net/profile/Alfonso_Martinez8/publication/267028000_Modelo_de_Procesos_para_la_Industria_de_Software_MoProSoft/links/544183840cf2e6f0c0f62d0d/Modelo-de-Procesos-para-la-Industria-de-Software-MoProSoft.pdf.

14. **"MSW". 10 de Abril de 2017,** de http://msw.com.mx/.

15. **"The Standish Group". 12 de Julio de 2016,** de http:// http://www.evergreenpm.com/standish-group-2015-chaos-report/.

16. **Pan, J. (1999).** Software testing. *Dependable Embedded Systems*, *5*, 2006.

17. **Roger S. Pressman.** Software Engineering: A Practitioner's Approach. Mcgraw-Hill Interamericana editores, s.a. De C.V. 2015. Octava Edición. ISBN: 0078022126

18. **"TMMi". 12 de Abril de 2017,** de https://www.tmmi.org/wp-content/uploads/2016/09/TMMi-Framework-R1.0-Spanish.pdf

19. **Van Veenendaal, E., & Wells, B. (2012).** Test maturity model integration (TMMi). Uitgeverij Tutein Nolthenius.

20. **Vega, K. C., & Dávila, A. (2014).** Evaluación teórica de la capacidad de procesos de Rational Unifi ed Process respecto del MoProSoft. Industrial Data, 13(2), 083-091.

21. **Von Wangenheim, C. G., Hauck, J. C. R., Salviano, C. F., & von Wangenheim, A. (2010, May).** Systematic literature review of software process capability/maturity models. In Proceedings of International Conference on Software Process Improvement and Capabity Determination (SPICE), Pisa, Italy.