Revista Industrial Data 20(1): 139-148 (2017)
DOI: https://doi.org/10.15381/idata.v20i1.13290
ISSN: 1560-9146 (Impreso) / ISSN: 1810-9993 (Electrónico)
Facultad de Ingeniería Industrial - UNMSM

Desarrollo de un modelo de pruebas funcionales de software basado en la herramienta SELENIUM

Recibido: 16/03/2017 Aceptado: 09/06/2017

Evelyn Chinarro Morales[1]
María Elena Ruiz Rivera[2]
Edgar Ruiz Lizama[3]

RESUMEN

Este proyecto se enfoca en presentar un modelo de referencia tomando herramientas ya creadas para automatizar el proceso de realización de las pruebas funcionales durante la fase de evaluación de la calidad del producto desarrollado. Esta automatización contempla la evaluación de ciertas herramientas automatizadas de administración de pruebas, así como la automatización de pruebas funcionales.

El desarrollo del software y el logro de un servicio informático de primera calidad es lo que actualmente se logra con las pruebas funcionales hechas en la etapa de desarrollo. Este proceso es una parte muy importante y crítica dentro del proceso de desarrollo de software, y debe realizarse con la mayor eficacia y la mejor eficiencia. Durante el proyecto se conocieron los procesos actuales que forman parte de la metodología de desarrollo de aplicaciones, tanto teórica como práctica. Se evaluaron herramientas y metodologías para automatización de pruebas. Se consideró muy importante que las soluciones tanto metodológicas como técnicas apoyen aspectos como: encontrar defectos en fases más tempranas del desarrollo, lograr mayor y mejor cobertura de funcionalidad durante las pruebas, la ejecución de las pruebas viables en costo y tiempo.

Palabras clave: Pruebas de software, casos de prueba, automatización de pruebas.

Development of a Functional Software Test Model Based on the SELENIUM Tool

ABSTRACT

This project focuses on presenting a reference model based on tools already created to automate the process of performing functional tests during the evaluation of the quality of the developed product. This automation includes the evaluation of certain automated test management tools and automated functional testing.

Software development and computer provide quality service is what is currently achieved with functional tests made in the development stage. This process is very important and critical part in the development process and must be done with greater efficiency and better efficiency. During the project the current processes that are part of the application development methodology, both theoretically and in practice met. Automation tools and methodologies for testing were evaluated. It was considered very important that both methodological solutions and technical support aspects such as: find defects in early stages of development, achieve greater functionality and better coverage during testing, implementation of viable testing cost and time.

Keywords: Software testing, test cases, test automation.

1. INTRODUCCIÓN

El proyecto consiste en implementar un modelo de referencia sobre las pruebas funcionales que se realizan en la empresa IT&B Consulting debido que actualmente esta tarea se realiza manualmente, esto origina que cada día los usuarios presenten diversidad de incidencias causadas por las fallas y/o errores no encontradas durante la etapa de pruebas realizadas al producto antes de pasar a producción.

Actualmente, el área de Control de Calidad (QA) de la empresa realiza las pruebas funcionales de manera manual y si se toma en cuenta el poco conocimiento que se tiene del negocio del cliente, se tiene como consecuencia que la empresa debe invertir más tiempo y recursos para subsanar los errores no encontrados. Es así como la empresa IT&B Consulting manifiesta el interés de mejorar su metodología de testing funcional e incorporar el uso de herramientas para automatización de pruebas funcionales a fin de optimizar el tiempo dedicado a las pruebas y así garantizar la entrega de un producto de calidad a sus usuarios.

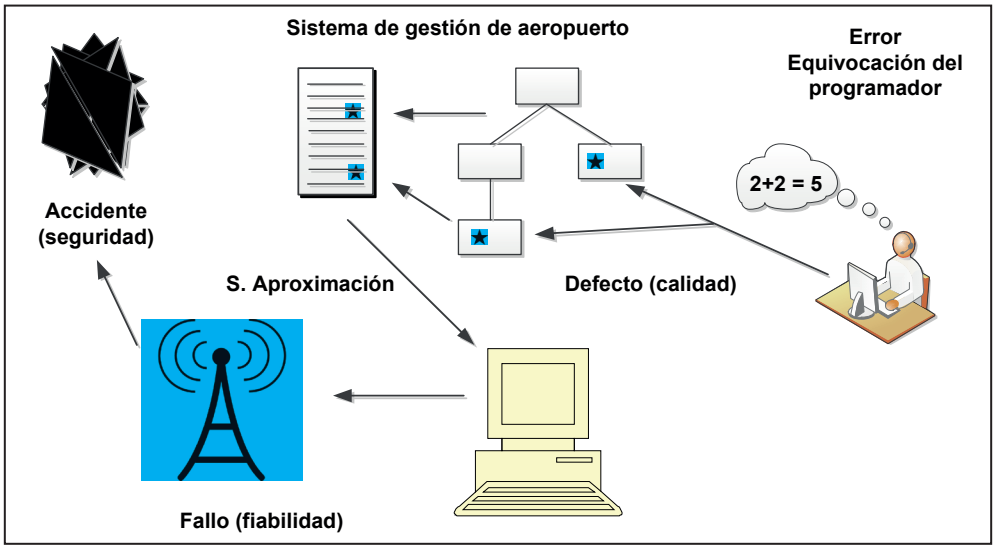
2. PLANTEAMIENTO DEL PROBLEMA

2.1. Antecedentes del Problema

Los sistemas o productos de software son creados por seres humanos por ende no se puede garantizar que no se hayan cometido errores durante la implementación de los mismos. Asimismo, el costo asumido por los errores encontrados en el sistema, luego de haberse adquirido, supone pérdida de dinero y tiempo de los recursos asignados para el proceso de desarrollo, es así que se inició con las pruebas de software para garantizar la disminución de fallos del sistema en producción.

Anteriormente las pruebas de software estaban orientadas a la corrección directa del código fuente de los programas. Éstas eran realizadas directamente por los programadores como se ve en la figura 1 (Panel Testing Centro de Excelencia, 2016).

1 Bachiller en Ingeniería de Software, UNMSM. Asistente de proyectos-QA en la empresa IT&B Consulting SAC.
E-mail: evelyn.chinarro@itbconsulting.pe
2 Licenciada en Computación. Profesora Asociada de la Facultad de Ingeniería de Sistemas e Informática, UNMSM.
E-mail: mruizr@unmsm.edu.pe
3 Magíster en Informática PUCP. Ingeniero Industrial UNMSM, Profesor Principal del Departamento de Ingeniería de Sistemas e Informática DAISI, UNMSM.
E-mail: eruizl@unmsm.edu.pe



Fuente: Tomado y adaptado de (Panel Testing Centro de Excelencia, 2016).

Figura 1. Pruebas funcionales - Antecedentes.

Conforme se ha ido avanzando en el tiempo y con el avance de la tecnología se inició la utilización masiva de tests para garantizar el cumplimiento con la especificación de los requerimientos dados por el usuario, dichos test se realizan al final del desarrollo del software. Estos tests se convierten en Casos de Prueba que se aplican a los productos desarrollados para encontrar errores y corregirlos.

En la actualidad, se propone y describe una metodología que integra análisis, revisión y testing en el ciclo de vida del desarrollo de software. Las pruebas de Software empiezan a integrarse en las diferentes metodologías del desarrollo. El objetivo general de las pruebas de software es confirmar o asegurar la calidad de los sistemas.

Las pruebas de software son un elemento crítico para la garantía de la calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación. Para implantar con éxito una estrategia de prueba de software es necesario abordar los puntos siguientes: especificar los requisitos del producto de manera cuantificable mucho antes que comiencen las pruebas, establecer los objetivos de la prueba de manera explícita, comprender que usuarios van a manejar el software y desarrollar un perfil para cada categoría de usuario, desarrollar un plan de pruebas, llevar a cabo revisiones técnicas formales para evaluar la estrategia de prueba y los propios casos de prueba (Gelperin, & Hetzel, 1988).

Existen diferentes tipos de pruebas de software asociadas a determinadas técnicas que establecen el grado de profundidad en el cual se diseñarán, ejecutarán y evaluarán teniendo como referencia la estructura interna de la aplicación. Las principales técnicas son: Caja Blanca, Caja Gris, Caja Negra, Pruebas de Unidad y las Pruebas del Sistema, que está constituida por una serie de pruebas diferentes (prueba de recuperación, prueba de seguridad, prueba de resistencia y prueba de rendimiento).

Dado lo expuesto, se denota que la etapa perteneciente a las pruebas de software es una parte fundamental en el proceso de desarrollo de software así que se debe considerar un tiempo definido para su ejecución además de definir la metodología o herramienta de apoyo a utilizar para así asegurar la más óptima ejecución de las pruebas funcionales.

2.2. Definición del problema

Actualmente, el área de QA de la empresa IT&B Consulting realiza sus pruebas funcionales de manera manual y teniendo además poco conocimiento del negocio del cliente, esto ocasiona que la empresa tenga que invertir más tiempo y recursos para subsanar los errores no encontrados, a su vez ocasiona que los usuarios reporten incidencias cuya frecuencia retrasa el óptimo desarrollo de sus funciones.

2.3. OBJETIVOS

Objetivo principal

Desarrollar un modelo de pruebas funcionales durante el proceso de desarrollo de software basado en la herramienta Selenium®.

Objetivos secundarios

- Definir los pasos que se seguirán en el modelo de referencia para la automatización de las pruebas funcionales.
- Priorizar los casos más frecuentes que se dan en la organización para mapearlos y documentar la solución utilizando una herramienta que guarde los datos obtenidos.
- Documentar todos los procesos del negocio para facilitar la definición de los Casos de Prueba durante el desarrollo del aplicativo.
- Evaluar las diferentes herramientas para Pruebas Funcionales de Selenium®.

2.4. Justificación

Actualmente la ejecución de las pruebas funcionales se realiza manualmente, registrando las incidencias de manera manual en un documento MS-Excel®. Se mapean los errores y sólo se prueba funcionalmente los flujos conocidos del negocio, esto causa que el usuario reporte diversos errores por una mala ejecución de las pruebas funcionales que se realizan en la etapa de desarrollo.

Se decidió desarrollar un modelo de referencia para automatizar las pruebas funcionales con el fin de brindar apoyo en el área de QA de la empresa Consultora. Este modelo permitirá a la empresa mejorar su metodología, ahorrando tiempo y recursos, además de garantizar el buen funcionamiento del aplicativo ya que en la etapa de desarrollo se mapearán la mayor cantidad de errores que pudiera tener el producto software (aplicativo desarrollado).

2.5. Propuesta metodológica

La solución involucra la creación y definición de un nuevo modelo para la realización de las pruebas funcionales al aplicativo antes de ser entregado al cliente. Adicionalmente se utilizará la herramienta Selenium® para automatizar las pruebas funcionales del aplicativo y detectar la mayor cantidad de defectos y a su vez, guardar un histórico de los casos de pruebas realizados y evaluados para futuros mantenimientos del aplicativo desarrollado (Selenium Developers Group, 2016).

3. CONCEPTOS BÁSICOS DE PRUEBAS DE SOFTWARE

3.1. Base teórica de las pruebas

Las pruebas se clasifican mediante tipos de pruebas y estos tipos están relacionados con respecto a la función que cumplen. Es importante tener en claro la clasificación ya que con esto se tiene un mejor conocimiento de la funcionalidad, es decir; para que se utilizan y cuando emplear un tipo de prueba especifico (Rodríguez, 2016).

3.2. Gestión de pruebas

Para el desarrollo de pruebas de software no solo hace falta el conocimiento de cómo desarrollarlas, también se deben realizar otro conjunto de tareas; entre estas tareas se encuentran la planificación de pruebas, la generación de casos de uso, el seguimiento de los errores, manejar las configuraciones del sistema y otras tareas. La realización de todas esas tareas es lo que se denomina gestión de pruebas dado que la mayoría de las veces son supervisadas por un individuo y realizadas por el grupo de trabajo. Para el caso del presente artículo se analiza las tareas de gestión de errores, generación de casos de uso, planificación de las pruebas y el manejo configuraciones a fin de establecer una herramienta automatizada de apoyo a la gestión de pruebas de software (Calidad y Software, 2016).

3.3. Pruebas funcionales

Se denominan pruebas funcionales o Functional Testing, a aquellas que tienen por finalidad: 1. Identificar inconsistencias, 2. Asegurar requisitos funcionales, 3. Reducir costoss de no conformidades, 4. Evitar reprocesos, 5. Mejorar la productividad, y 6. Aumentar la satisfacción del cliente, (Quality, 2016). Las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados, es común que este tipo de pruebas sean desarrolladas por analistas de pruebas con apoyo de algunos usuarios finales, esta etapa suele ser la última etapa de pruebas y al dar conformidad sobre esta, el paso siguiente es el pase a producción.

A este tipo de pruebas se les denomina también pruebas de comportamiento o pruebas de caja negra, ya que los testers o analistas de pruebas, no enfocan su atención a como se generan las respuestas del sistema, básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas.

Las pruebas son los procesos de ejecución de un programa en los que se intenta descubrir errores. Esto supone un cambio de punto de vista; nos cambia la idea que tenemos de que una prueba es exitosa si no consigue errores. Para poder diseñar ca-

sos de prueba que proporcionen un buen resultado, el ingeniero de software debe tener en cuenta los principios de las pruebas (Rodríguez, 2016).

Existen diferentes técnicas de pruebas funcionales como por ejemplo las Prueba Caja Blanca y Caja Negra.

3.4. Herramientas de apoyo

Como se sabe que la etapa de pruebas puede llegar a requerir una cantidad de tiempo y esfuerzo incluso mayor que el de la de creación del software; en la actualidad se encuentran disponibles diferentes herramientas que sirven de soporte a la realización de dichas pruebas. Cuando se habla de herramientas que sirven de soporte ó apoyo se habla de herramientas que permiten el desarrollo de pruebas automatizadas y el diseño de pruebas, tomando en cuenta esta funcionalidad de una herramienta estas pueden clasificarse en herramientas de apoyo a la gestión de pruebas y/o al desarrollo de pruebas de software (Calidad y Software, 2016).

Actualmente existen herramientas que ofrecen soporte en ambos campos por lo tanto se pueden tener herramientas que ofrezcan soporte completo para la etapa de pruebas en el desarrollo de software, esto dependiendo si las pruebas a las cuales dicho software da apoyo son las mismas que en un momento dado se necesitan aplicar. Un ejemplo de una herramienta de apoyo tanto para el desarrollo como para la gestión de pruebas es la herramienta JUnit, la cual proporciona soporte para las pruebas unitarias en el ambiente de desarrollo de software (Calidad y Software, 2016).

Selenium

Selenium es un conjunto de utilidades que facilita la labor de obtener juegos de pruebas para aplicaciones web, además es multiplataforma, pudiendo instalarse en cualquier sistema operativo. Para ello Selenium permite grabar, editar y depurar casos de prueba, que pueden ser ejecutados de forma automática e iterativa posteriormente, (Selenium Developers Group, 2016).

Además de ser una herramienta para registrar acciones, permite editarlas manualmente o crearlas desde cero. Las acciones se basan en el uso de diferentes API's en diferentes lenguajes (PHP, Ruby, JAVA, Javascript, etc). Entre sus principales características se puede mencionar:

- Facilidad de registro y ejecución de los test.
- Referencia a objetos DOM en base al ID, nombre o a través de XPath.
- Auto-completado para todos los comandos.
- Las acciones pueden ser ejecutadas paso a paso.
- Herramientas de depuración y puntos de ruptura (breakpoints).
- Los test pueden ser almacenados en diferentes formatos.

El potencial de esta herramienta puede ser utilizado para la grabación de las pruebas funcionales durante la Generación de pruebas de regresión. Con este servicio se consigue obtener una batería de pruebas automatizadas que podrán ser utilizadas cuando sea necesario repetir las pruebas.

Selenium IDE

Permite crear Scripts de pruebas de manera rápida ya que trabaja como una "Grabadora" que captura todas las acciones que se hacen sobre la interfaz de usuario (una especie de macro),generando automáticamente los comandos del Script de pruebas (Selenium Developers Group, 2016). Esta herramienta trabaja como un "complemento" (add-on) de Firefox, el cual permite Grabar y Reproducir de manera simple las iteraciones con el browser.

Otras Características:

- Manejo inteligente de la identificación de Campos o Elementos en la página Web, se puede usar IDs, nombres o XPath.
- Autocompleta todos los comandos de Selenium
- Debug y Puntos de quiebre.
- Permite grabar los Scripts en formato HTML, Ruby, JUnit, entre otras.
- Soporte para el archivo "user-extensions.js", el cual permite agregar funciones javascript personalizadas.
- Opciones para realizar validaciones automáticas del título para cada página en la que se esté navegando (Junta de Andalucía, 2016).

4. MODELOS RELACIONADOS A PRUEBAS

4.1. Método para generar casos de pruebas funcionales

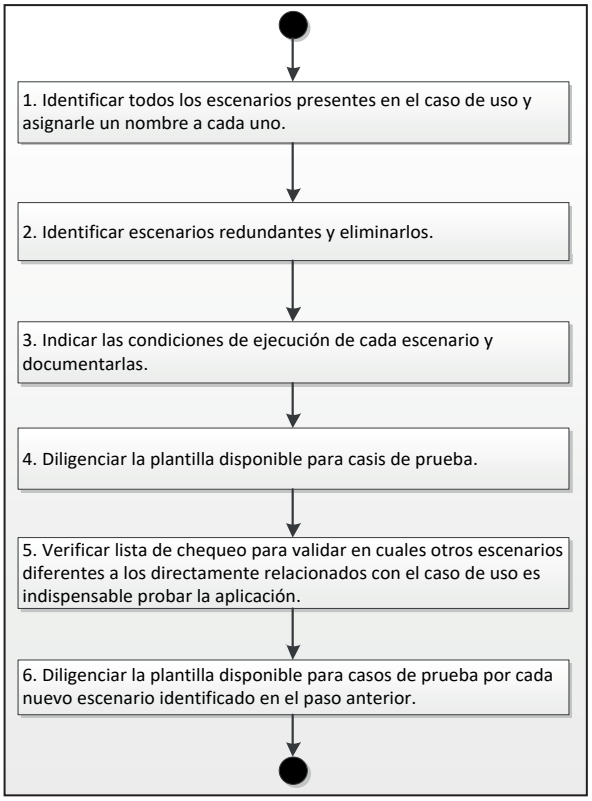
Las pruebas de software se dividen en dos grandes grupos, en este artículo se tendrá en cuenta las funcionales, las cuales se aplican al producto final, luego de la fase de desarrollo, éstas permiten detectar cuáles son los puntos que no cumplen los requerimientos dados por el usuario además que

puedan fallar en cuanto a la funcionalidad desarrollada (González, 2009).

Para generar los casos de pruebas funcionales de acuerdo a la nueva metodología primero se debe tener en cuenta la información de entrada que se necesita para poder obtener los casos de prueba a ejecutar. La información de entrada son: 1. Casos de uso del sistema y plantillas de descripción. 2. Lista de verificación de aspectos adicionales a chequear en pruebas funcionales. 3. Plantilla caso de prueba. 4. Aplicación lista para probar. Siendo la salida: Casos de prueba listos para ejecución por el tester, (González,, 2009).

Uno de los inputs es la lista de verificación de los aspectos adicionales que se deben tener en cuenta para las pruebas funcionales, esto permitirá saber si se probaron los casos críticos para el aplicativo, otro punto son los casos de uso del sistema y las plantillas con la información y descripción de cada uno de ellos.

Como se muestra en la figura 2 (González, 2009), se describe paso a paso la metodología propuesta, en el **paso 1** se describe que se debe identificar los escenarios relevantes y correspondientes a los flujos normales, de error o alternativos que sucedan al ejecutar el caso de prueba.



Fuente: Tomado de (González, 2009).

Figura 2. Proceso para creación de los casos de pruebas funcionales.

En el **paso 2** es tener en cuenta los escenarios ya identificados en el paso 1 y eliminarlos si es que se encontrarán repetidos y validados en otro escenario. En cuanto al **paso 3**, cada escenario que se tenga luego de realizar la eliminación de los repetidos será un caso de prueba.

En el **paso 4** se hace referencia a documentar toda la información requerida en la plantilla de los casos de prueba que se están identificando.

En el **paso 5** de tendrá en cuenta las condiciones para ejecutar los casos de prueba, los valores necesarios para poder probar el buen funcionamiento del aplicativo por ejemplo los campos adicionales, los campos obligatorios, los tipos de datos, etc.

Por último, en el **paso 6** se documenta en la plantilla las condiciones identificadas en el paso anterior para completar la información requerida por la plantilla. Ver figura 2.

Se observa que si bien es cierto que la planificación y diseño de las pruebas funcionales en la primera etapa de desarrollo del proyecto ayuda a comprobar y verificar que se cumplan los requerimientos dados por el usuario.

4.2. Proceso de desarrollo enfocado en las pruebas del software

Se describe una a una las fases del desarrollo desde el inicio del proceso que comienza cuando el usuario acepta la oferta para iniciar el desarrollo del software. Es así que se consideró importante definir uno a uno al detalle los procesos que se siguen para el desarrollo de las pruebas para así lograr la satisfacción del usuario al entregar la versión definitiva del software y no luego de que ya se hizo la entrega final es decir cuando ya está en producción ya que puede ocasionar el descontento del usuario al encontrarse defectos en la entrega final, por eso es mejor aplicar las pruebas del software y las estrategias de las mismas durante la etapa de desarrollo del software (Prado, 2007).

Se empieza por describir la parte inicial del proceso que según definición es cuando el cliente acepta que se empiece con el desarrollo del software para ésta parte se debe tener en consideración desde ya las pruebas de software que se aplicarán durante el proceso.

Se disponen de diversas metodologías y estándares que apoyarán a las pruebas de software esto en cuanto a la parte inicial. En la parte de Planificación se definirán las técnicas de pruebas a usar, si se apoyará en alguna herramienta, criterios para los casos de prueba, todo esto se debe definir en el

Plan de Pruebas. Para el cual se define el alcance de la aplicación, la plataforma donde se probará, quiénes realizarán las pruebas, etc.

Hay consideraciones importantes en esta etapa como que las pruebas deben estar presentes a lo largo de todo el cicla de vida del desarrollo según el modelo en V (Sarco, 2016). A continuación, siguiendo con el ciclo de vida se pasa a la etapa de especificación de los Requisitos donde se verifica que cada característica pueda ser validada es decir probada. La revisión de los requisitos o requerimientos permite detectar errores en las fases tempranas evitando que se genere un costo extra que se detectará al final en la entrega del software.

Se continúa con la etapa de Arquitectura y Diseño, dentro de esta fase se tiene la construcción del aplicativo en donde se realiza las revisiones de código, como las pruebas unitarias. Esto permitirá tener un código de alta calidad y que se cumpla con los estándares, para esto se puede apoyar de herramientas de análisis de código.

Luego se llega a la fase más importante, que es la fase de pruebas en donde se tiene diferentes niveles y tipos de pruebas, siendo estos niveles (de abajo hacia arriba) los siguientes: 1. Testing de componentes, 2. Testing de integración, 3. Testing del sistema, y 4. Testing de aceptación, las que se pueden aplicar al software, (Sarco, 2016).

A continuación, se tiene las pruebas de integración, que es donde se comprueba la funcionalidad de las interfaces entre las distintas partes que componen un sistema. A su vez están las pruebas de Validación, son aquellas que se realizan sobre un software integrado para evaluar el cumplimiento de los requerimientos. Las pruebas de Sistema que evalúan el sistema ya integrado teniendo en consideración el rendimiento, la resistencia, la seguridad y la usabilidad.

Por último tenemos las pruebas de Aceptación las cuales se realizan con el usuario en donde se determina si el sistema desarrollado cumple con lo requerido y se obtiene la conformidad del cliente.

4.3. Pruebas de software en el ciclo de vida del software

Al ver la situación de las empresas que tienen que buscar a testers que tengan bastante conocimiento de las pruebas que realizan al software para que cuando se haga entrega del mismo al cliente, éste no encuentre errores de funcionamiento durante su uso (Serna & Arango, 2012).

Se describen los posibles problemas que puedan encontrar los testers al momento de realizar el proceso de las pruebas, así como se describen 4 fases donde se agruparán los problemas relacionados y que se deben de resolver antes de pasar a la siguiente fase.

FASE 1: MODELAR EL ENTORNO DE SOFTWARE, consiste en simular la interacción entre el aplicativo y su entorno, para ello se debe identificar y simular las interfaces que va a utilizar el aplicativo. Las interfaces comunes que se utilizan son: las interfaces humanas, incluyen los métodos con los cuales las personas se comunican con el aplicativo, por ejemplo: los clics del ratón, pulsaciones del teclado, etc.

Los testers deciden como organizar los datos para comprender como los juntara para realizar una prueba efectiva. Las interfaces de software, las cuales indican como utiliza el software al sistema operativo, la base de datos o la librería, en tiempo de ejecución. Tenemos también las interfaces del sistema de archivos, siempre y cuando el sistema desarrollado lea o escriba datos en archivos externos.

FASE 2: SELECCIONAR ESCENARIOS DE PRUEBA, los casos de prueba cuestan tiempo y dinero, el conjunto de casos de prueba que muchos testers eligen son los que cumplan con las metas de cobertura, por ejemplo cobertura de código fuente, declaraciones de código o la cobertura de entradas. En esta fase los testers se interesan más por los caminos de ejecución, secuencias de declaraciones de código que representan un flujo del aplicativo, se busca el conjunto de escenarios que garantizara encontrar la mayoría de los errores. Así mismo los escenarios de pruebas pueden asegurar que el software funciona de acuerdo a los requerimientos especificados.

FASE 3: EJECUTAR Y EVALUAR LOS ESCENARIOS, los testers convierten los escenarios en formatos ejecutables, de modo que resulten similares la acción que realizaría un cliente típico, los escenarios de prueba se ejecutan manualmente, debido a esto los testers están buscando la forma de automatizarlos, se requiere la simulación de cada input y el destino de la salida de todo el entorno. La evaluación implica la comparación de las salidas obtenidas durante la ejecución de los escenarios de prueba con respecto a la salida real del software.

FASE 4: MEDIR EL PROGRESO DE LAS PRUEBAS, consiste en contar las cosas, el número de entradas aplicadas, el porcentaje de todo el código revisado, el nuero de veces que se ha utilizado el aplicativo, el número de veces que se obtuvo respuesta con éxitos, etc.

Los testers deben determinar cuándo detener las pruebas o cuando está listo el aplicativo para entregarlo al cliente. Se necesita medidas cuantitativas de los errores encontrados para disminuir la probabilidad de que el usuario los encuentre cuando este en producción.

4.4. Herramientas de apoyo

4.4.1. Automatización de las pruebas funcionales con herramientas open source

Para realizar la automatización de las pruebas hay diversas herramientas de apoyo así tenemos: las pruebas estáticas, herramientas para la revisión del análisis y el modelado; herramientas para el diseño de las pruebas; herramientas para la ejecución de los casos de prueba y por último las herramientas para la carga y monitoreo, (Esmite, Farías, Farías, & Pérez, 2007).

Actualmente en el Centro de Ensayos de Software (CES) utilizan un proceso llamado ProTest el cual es un proceso de testing funcional independiente el cual se describe una extensión del proceso, así como nuevas actividades específicas para la automatización.

Se apoyó en las herramientas del grupo Selenium (Selenium Core, IDE y Remote Control) (TestNBug in Automation Basics, 2016), para la ejecución de las pruebas, éstas herramientas permiten crear y ejecutar pruebas automatizadas sobre aplicativos web.

Se analizó la situación de la entidad llamada “Centro de Ensayos de Software” (CES) la cual agrupa la mayoría de empresas productoras de software de Uruguay las cuales ofrecen diversos servicios tales como: servicios de prueba independiente, consultoría y capacitación.

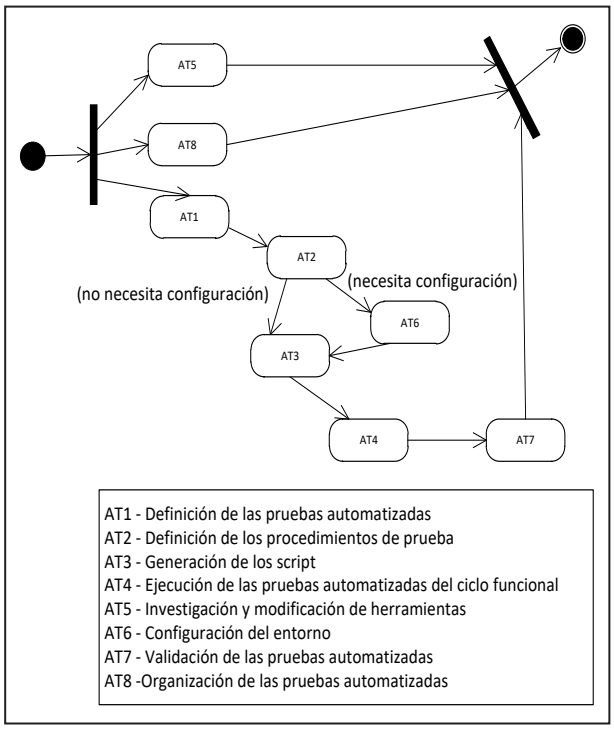
Se define la metodología utilizada tiene una serie de pasos y/o actividades definidas en la figura 5 se comienza con la actividad AT1 la cual trata de la definición de las pruebas automatizadas, actividad donde se define las funcionalidades que se probarán con las pruebas automatizadas. Ver figura 3.

En la AT2 tenemos la definición de los procedimientos de prueba, en la cual se definen los casos de pruebas y los scripts que conformarán las pruebas automatizadas. Se debe especificar cada paso que se realizará y las verificaciones que se realizarán durante las pruebas.

Se continua con la AT3, generación de casos de pruebas y los scripts, se obtiene los casos de pruebas con sus scripts correspondientes así mismo se verifica la correcta funcionalidad del script ejecutado.

En el AT-4, ejecución de las pruebas automatizadas del ciclo funcional, el objetivo principal es ejecutar una prueba completa de los casos de pruebas correspondientes al ciclo funcional y verificar que se ejecute correctamente en el entorno de pruebas preparado para dicho fin. En la metodología propuesta se utilizará herramientas open source, las cuales se necesitan investigar, eso se da en el paso AT-5, donde se investiga y /o se modifican las herramientas que se están utilizando para la automatización.

En la actividad AT-6 se configura el entorno en el cual se van a desarrollar las pruebas funcionales automatizadas. Como penúltimo paso tenemos la validación de las pruebas automatizadas en el entorno del usuario, se realizan las pruebas y se verifican que los scripts y los casos de pruebas están funcionando correctamente. Por último en la actividad AT-8 se organizan y gestionan los artefactos obtenidos de las pruebas funcionales automatizadas.



Fuente: Recuperado de (Esmite, Farías, Farías, & Pérez, 2007).

Figura 3. Diagrama de actividad de la metodología propuesta.

5. APORTE

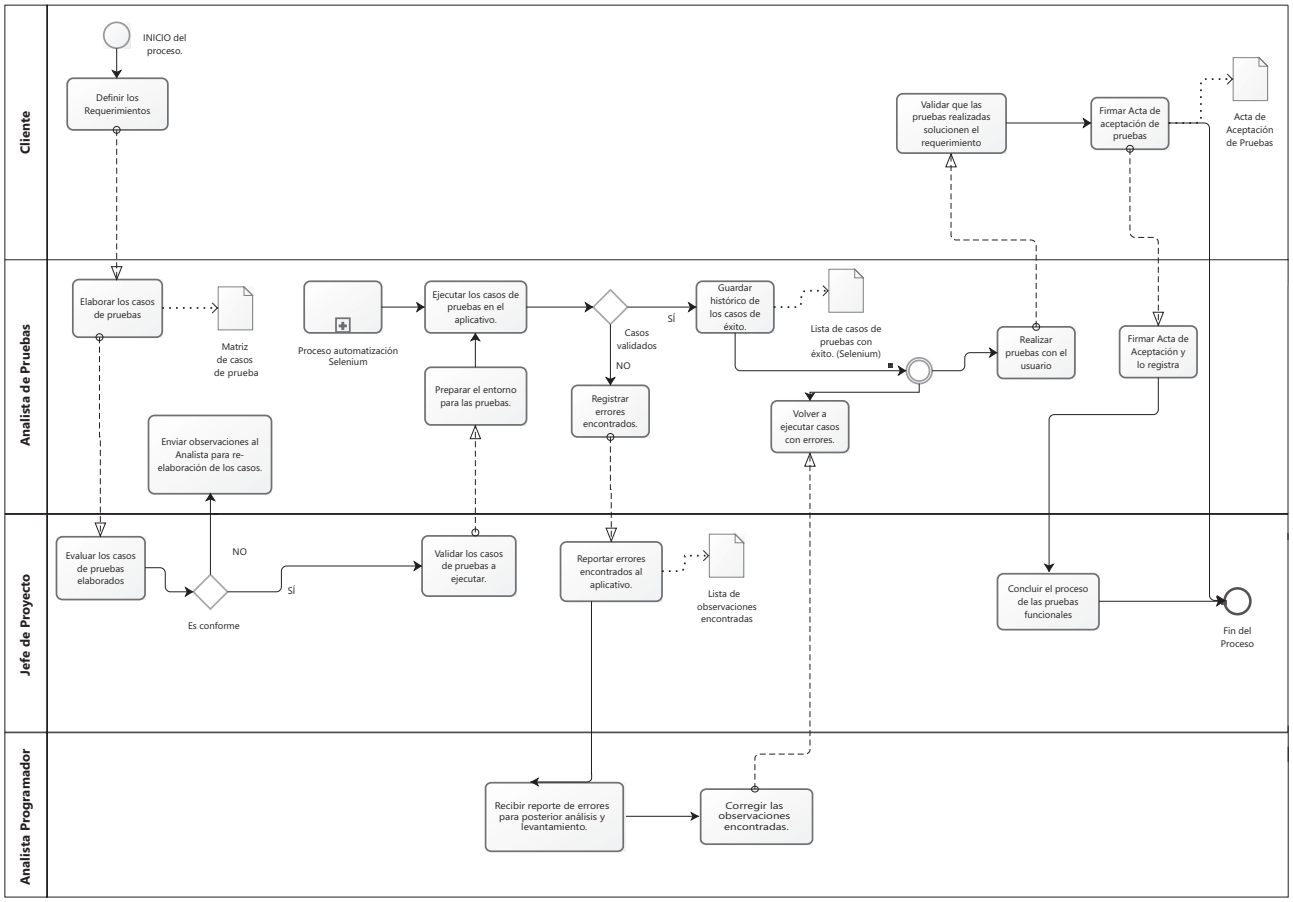
Se creará un nuevo modelo, basado en los modelos estudiados y descritos en el estado de arte, donde se describirá paso a paso las actividades a realizar durante el proceso de desarrollo de las pruebas

funcionales, en donde se utilizará la herramienta Selenium para aprovechar las funcionales que éste contiene y así automatizar las pruebas funcionales.

En el modelo creado, como se muestra en la figura 6, tiene como primer paso la definición de los requerimientos, ésta actividad se debe desarrollar con el usuario ya que esta etapa es primordial para pasar a la siguiente fase que es el análisis. Asimismo, en esta actividad se debe tener el entendimiento claro del flujo del negocio, lo cual es necesario, ya que nos permitirá determinar los posibles escenarios donde se realizarán los casos de prueba. Al respecto ver figura 4.

Seguidamente, luego de determinar los posibles escenarios, se procede a elaborar los casos de prueba, con el conocimiento de los requerimientos y aspectos vitales del negocio, se elabora una matriz de casos de pruebas funcionales.

Para ejecutar los casos de pruebas creados, primero el jefe de proyecto debe evaluar los casos y si éstos satisfacen los requerimientos dados por el usuario. De encontrar alguna inconformidad u observación se informa al analista de pruebas para que evalúe y re-diseñe el caso de prueba con las observaciones encontradas. Si al evaluar los casos de pruebas, el jefe de proyecto da su conformidad, se prepara el entorno, similar al que se tendrá cuando el software pase a producción, para poder iniciar con la ejecución de los casos de prueba.



Fuente: Elaboración propia.

Figura 4. Modelo de referencia propuesto.

Al comenzar a ejecutar los casos de prueba, primero se procede con el subproceso de automatización de las pruebas funcionales el cual se apoyará con la herramienta Selenium, ya que contiene diversas funcionalidades necesarias para una óptima realización de las pruebas funcionales al aplicativo. La automatización consiste en la grabación de las pruebas funcionales durante la Generación de las pruebas. Con esto se consigue obtener un conjunto de pruebas automatizadas que podrán ser utilizadas cuando sea necesario para repetir las pruebas, es decir guarda un histórico de las pruebas para una reutilización de ser necesaria.

Al terminar de ejecutar los casos de pruebas, se presentan dos casos, primero los casos que se prueban con éxito y se llega al resultado esperado, y segundo los casos que no logran obtener la respuesta positiva. Se procede a reportarlo y a evidenciarlo para que se corrija, se hace del conocimiento al programador para que subsane los errores encontrados. Luego de que el programador solucione las observaciones encontradas, el analista de pruebas vuelve a ejecutar los casos de prueba que no tuvieron éxito o fallaron durante el proceso.

Al finalizar, la siguiente actividad es realizar las pruebas con el usuario en su ambiente para luego firmar con él el Plan de Aceptación de Pruebas donde se deja constancia que los involucrados están conformes con las pruebas realizadas y que se cumplen los requerimientos solicitados, ya que éste documento valida el correcto despliegue de los casos de prueba y la satisfacción del usuario con el aplicativo desarrollado.

CONCLUSIONES

- Con el modelo de pruebas funcionales propuesto se garantiza la disminución del tiempo de pruebas, así como la optimización de los recursos a utilizar.
- Se logró definir y automatizar los pasos a seguir según el modelo desarrollado para garantizar un mejor desempeño durante el proceso de las pruebas funcionales en la etapa de desarrollo del aplicativo.
- Las incidencias reportadas por los usuarios en producción disminuyen en un porcentaje considerable y mejora el proceso de las pruebas.

REFERENCIAS BIBLIOGRÁFICAS

[1] Calidad y Software. (15 de marzo de 2016). *Calidad y Software*. Recuperado de: http://www.calidadysoftware.com/

[2] Esmite,I., Farías, M., Farías, N., & Pérez, B. (s.f.). Automatización y gestión de las pruebas funcionales usando herramientas open source. (X. C. Computación, Ed.) *IV Workshop de Ingeniería de Software y Bases de Datos*, 294 - 305.

[3] Gónzales. (2009). Método para generar casos de prueba funcional en el desarrollo de software. *Revista Ingenierías* - Universidad de Medellín, 8(15), 29 - 36. Recuperado de: https://goo.gl/P3SJT6

[4] Gelperin,D., & Hetzel, W. (June de 1988). The growth of software testing. (ACM, Ed.) *Comunications of the ACM, 31*(6), 687 - 695

[5] Junta de Andalucía. (10 de 03 de 2016). *Selenium y la automatización de las pruebas*. Recuperado de: https://goo.gl/svqmof

[6] Panel testing centro de excelencia. (10 de marzo de 2016). *Panel Testing - Centro de Excelencia, Autor en panel sistemas*. Recuperado de: https://goo.gl/EUg8Tu

[7] Quality, Y. y. (Ed.). (11 de Marzo de 2016). *VyV-Quality*. Recuperado de: https://goo.gl/PwFQ4r

[8] Rodriguez (2 de abril de 2016). *Errores comunes al hacer puebas de sopftware de nuestro codigo*. Recuperado de: https://goo.gl/CChqs4

[9] Raja Prado. (2007). Casi todas las pruebas del software. *Actas de talleres de ingeniería del software y bases de datos, 1*(4), 43 - 46.

[10] Sarco,(15 de marzo de 2016). *Testing en español*. Recuperado de: https://goo.gl/V6wFBw

[11] Selenium Developers Group. (14 de marzo de 2016). *Selenium Web Browser Automation*. (Selenium Automates Browsers) Recuperado de: http://www.seleniumhq.org/projects/

[12] Serna,Edgar, & Arango, F. (2012). Prueba del software: más que una fase en el ciclo de vida. *Revista de Ingeniería*, 34 - 40.

[13] TestNBug in Automation Basics. (12 de Marzo de 2016). *Automation 6esting - TestNBug*. Recuperado de: https://goo.gl/U4wWKF