TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE APIZACO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

“MODELO PARA LA GENERACIÓN Y EJECUCIÓN DE PRUEBAS COMO MEDIO DE VERIFICACIÓN Y VALIDACIÓN DE PRODUCTOS DE SOFTWARE DE CALIDAD”

T E S I S

PARA OBTENER EL GRADO DE:

MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:

LIC. SAID PÉREZ FLORES

DIRECTORES:

M. en C. JOSÉ JUAN HERNÁNDEZ MORA

M. en C. MARÍA GUADALUPE MEDINA BARRERA

APIZACO, TLAXCALA **DICIEMBRE 2016**

SEP

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Apizaco

Apizaco, Tlax., 08 de Agosto de 2016

ASUNTO: **Aprobación del trabajo de Tesis de Maestría.**

DR. JOSE FEDERICO CASCO VASQUEZ
JEFE DE LA DIVISION DE ESTUDIOS DE
POSGRADO E INVESTIGACION,
P R E S E N T E.

Por este medio se le informa a usted, que los integrantes de la **Comisión Revisora** para el trabajo de tesis de maestría que presenta el **LIC. SAID PEREZ FLORES** con número de control **M09370370**, candidato al grado de **Maestro en Sistemas Computacionales** y egresado del **Instituto Tecnológico de Apizaco,** cuyo tema es **"MODELO PARA LA GENERACION Y EJECUCION DE PRUEBAS COMO MEDIO DE VERIFICACION Y VALIDACION DE PRODUCTOS DE SOFTWARE DE CALIDAD"**, fue:

A P R O B A D O

Lo anterior, al valorar el trabajo profesional presentado por el candidato y constatar que las observaciones que con anterioridad se le marcaron así como correcciones sugeridas para su mejora ya han sido realizadas.

Por lo que se avala se continúe con los trámites pertinentes para su titulación.

Sin otro particular por el momento, le envió un cordial saludo.

LA COMISION REVISORA

M.C. JOSE JUAN HERNANDEZ MORA | **M.C. MARIA GUADALUPE MEDINA BARRERA**

M.C.C. MARIA JANAI SANCHEZ HERNANDEZ | **M.D.S. HIGINIO NAVA BAUTISTA**

C. p.- Interesado.

Apizaco, Tlax.; 09 de agosto de 2016

No. de Oficio: DEPI/251/16

ASUNTO: **Se Autoriza Impresión de Tesis de Grado.**

LIC. SAID PÉREZ FLORES,
CANDIDATO AL GRADO DE MAESTRO
EN SISTEMAS COMPUTACIONALES
No. de Control: **M09370370**
P R E S E N T E.

Por este medio me permito informar a usted, que por aprobación de la Comisión Revisora asignada para valorar el trabajo, mediante la Opción: **I Tesis de Grado por Proyecto de Investigación**, de la **Maestría en Sistemas Computacionales**, que presenta con el tema: **"MODELO PARA LA GENERACIÓN Y EJECUCIÓN DE PRUEBAS COMO MEDIO DE VERIFICACIÓN Y VALIDACIÓN DE PRODUCTOS DE SOFTWARE DE CALIDAD "** y conforme a lo establecido en el Procedimiento para la Obtención del Grado de Maestría en el Instituto Tecnológico, la División de Estudios de Posgrado e Investigación a mi cargo le emite la:

AUTORIZACIÓN DE IMPRESIÓN

Debiendo entregar un ejemplar del mismo debidamente encuadernado y seis copias en CD en formato PDF, para presentar su Acto de Recepción Profesional a la brevedad.

Sin otro particular por el momento, le envío un cordial saludo.

A T E N T A M E N T E
PENSAR PARA SERVIR, SERVIR PARA TRIUNFAR®

DR. JOSÉ FEDERICO CASCO VÁSQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN

Secretaría de Educación Pública
Instituto Tecnológico de Apizaco
División de Estudios de Posgrado
e Investigación

C.p.- Expediente.

JFCV/MJSH*mebr

No es verdad que las personas paran de

perseguir sueños porque se hacen viejos,

se hacen viejos porque paran de perseguir sus sueños.

Gabriel García Márquez.

A mi familia y a dios, por el apoyo incondicional en los aciertos

y errores a lo largo de este proceso,

por creer y confiar el mi durante todo el camino.

A mis amigos y profesores por los consejos brindados,

a todos ellos les dedico este trabajo por haber sido participes

para alcanzar un objetivo más en mi vida.

Agradecimientos

Agradezco a CONACYT por la confianza y el apoyo económico brindado a mi preparación.

Agradezco al Instituto Tecnológico de Apizaco por brindar las herramientas y conocimientos necesarios para realizar mis tareas, trabajos e investigaciones satisfactoriamente, que el día de hoy se reflejan en la culminación de un objetivo más en mi vida.

A mi comité tutorial, Maestro José Juan, Maestra María Guadalupe, Maestra María Janai y Maestro Higinio, por ser la guía del proyecto, por el apoyo, la paciencia y aportar sus conocimientos para poder cumplir este trabajo.

A todos los profesores de la maestría en sistemas computacionales del Instituto Tecnológico de Apizaco que han contribuido en mi formación académica a lo largo de este tiempo, brindando sus conocimientos y consejos para hacer de mi persona un mejor ser.

A mis compañeros de generación y otras áreas, a mis amigos incondicionales que estuvieron con migo en las buenas y en las malas, a todos ellos por el apoyo y consejos brindados, la enseñanza y compañerismo obtenido.

Agradezco infinitamente a mis padres, Piedad Flores y Marco Pérez por apoyarme, guiarme y confiar en mi durante todo el proceso, por sus sabios consejos, por creer en mis metas, objetivos y estar conmigo en los buenos y malos momentos, a mi hermana Xally por su cariño y atenciones, a mi hermano Rafael por su consejos y apoyo brindado, a mis abuelos, tíos, primos, y sobrinos por ser pilar de alegrías en mi persona y mantenerme el pie.

Resumen

El principal objetivo de la presente investigación ha sido la propuesta de una metodología para la generación y ejecución de pruebas como medio de verificación y validación de productos de software de calidad, que pueda ser implementada en todas aquellas empresas, organizaciones, e instituciones públicas y privadas para el cumplimiento de sus objetivos en el área de pruebas, sin perder de vista los elementos básicos que debe cumplir un modelo, conservando las buenas prácticas en las actividades, proporcionando instrumentos y herramientas para llevar a cabo el fin propuesto.

El presente trabajo está constituido de tres etapas fundamentales para la elaboración y ejecución de pruebas de software, detallado de la siguiente forma: en la primer fase se encuentra el diseño de pruebas, realizando actividades como la gestión de plantillas, identificación de requerimientos y casos de uso del sistema a liberar y la generación del plan de pruebas para guiar todo el proceso. La segunda etapa está determinada por la construcción y ejecución de los escenarios, estableciendo como metas principales la identificación, creación y aplicación de casos de prueba al sistema que se esté desarrollando. Finalmente, la tercera etapa está expresada por el seguimiento y resultados obtenidos en la etapa anterior, con tareas específicas en la verificación y validación de fallos encontrados, cierre y resultado final del proceso para generar el reporte final con la aprobación y liberación por parte del área de pruebas del software evaluado.

Para el análisis y resultados de la aplicación de la metodología, se evaluaron dos proyectos de software libre, con la ayuda de las plantillas y proceso presentado en la investigación. El resultado de los casos de estudio refleja una valoración en la funcionalidad, operatividad y rendimiento del software, verificando y validando el cumplimiento de estos factores en la liberación del producto, disminuyendo con esto, la liberación de productos con errores, logrando aumentar la calidad del producto para su publicación.

Abstrac

The main objective of this research has been the proposal of a methodology for the generation and execution of tests as a means of verification and validation of software products quality, which can be implemented in all companies, organizations, and institutions public and private for the fulfillment of its objectives in the test area, without losing sight of the basic elements that must meet a model, preserving the best practices in the activities, providing tools and resources to carry out the proposed order.

This work consists of three basic stages for the development and implementation of software testing, detailed as follows: in the first phase is the design of tests, performing activities such as managing templates, identification requirements and case release system use and generation of test plan to guide the process. The second stage is determined by the construction and implementation of scenarios, setting as primary goals the identification, development and implementation of test cases to the system being developed. Finally, the third stage is expressed by monitoring and results obtained in the previous stage, with specific tasks in the verification and validation of bugs found, closing and end result of the process to generate the final report with the approval and release by the area software testing evaluated.

For analysis and results of the application of the methodology, two projects of free software, with the help of templates and process presented in the research they were evaluated. The outcome of the study cases reflects an assessment on functionality, operability and performance of the software, verifying and validating compliance with these factors in product reléase, decreasing this, the release of products with errors, making increase the quality of product for publication.

Índice General

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

En la actualidad las pruebas del software juegan un papel muy importante en los procesos y actividades de la mayoría de las empresas y negocios, debido a la gran expansión que la ciencia y la ingeniería han tenido en esta última década. Hoy en día el software está relacionado con la mayoría de los ámbitos, por ejemplo: medicina, física, química, entretenimiento, procesos industriales, telecomunicaciones, etc. Permitiendo cumplir a las personas tareas comunes de manera sencilla, además de cubrir grandes necesidades a través de sistemas y aplicaciones. Lo que exige que la calidad y la confianza de los sistemas automatizados sea la adecuada ya que día a día muchos negocios, dependencias y personas físicas opten por expandir sus productos o servicios a través internet, llegando satisfactoriamente a una mayor cantidad de individuos de manera más simple o bien desarrollar procesos o actividades de forma sencilla y fácil.

Se ha sabido de muchos sucesos desfavorables que han hecho que la desconfianza en el uso de productos, sistemas o aplicaciones de software sea cada vez mayor, esto debido a que las empresas al utilizar nuevos productos y servicios de las tecnologías de la información en las diferentes áreas generen productos de baja calidad o con errores. Esto ha provocado que la demanda y el incremento en la calidad de los productos en una empresa desarrolladora de software sea fundamental para poder competir, adaptarse y sobrevivir en el mercado para satisfacer sus necesidades del cliente.

Por todo ello cabe reflexionar sobre el tipo de metodología, modelo, herramienta o estándar que se debe ocupar para la creación de pruebas que generen un desarrollo fiable y confiable a la hora de verificar y validar los productos de software. Así mismo si las metodologías convencionales cumplen y pueden satisfacer las necesidades cambiantes de los clientes y del mercado, sino también los requisitos el diseño, de la

tecnología a ocupar, del dispositivo a utilizar y muy importante de la forma en cómo deben generarse y ejecutarse las pruebas del software.

La presente investigación propone la construcción de un modelo para el desarrollo y la ejecución de pruebas funcionales, integrales y seguridad enfocadas al software que puedan cumplir con los objetivos de las diversas aplicaciones que este pueda tener incluyendo dispositivos móviles, aportando un guía que lleve de la mano al probador en estas dos actividades mencionadas desde un punto de vista metodológico y así mismo lograr que la versión de software final tenga un impacto favorable a los usuarios finales en cuanto a su contenido y funcionalidad.

1.1 Descripción del problema

Hoy en día el desarrollo del software es una de las actividad más complejas a la hora de tratar de solucionar problemas, facilitar tareas o crear aplicaciones para procesos específicos dentro de las actividades humanas, lo que lleva a tomar en cuenta el tiempo en el que se desarrollan y aplican ya que esto implica tiempo, dinero y esfuerzo para la empresa.

En la actualidad los errores y defectos que surgen en el software forman parte de una estadística muy importante que afecta a las empresas y usuarios finales, se ha determinado que un software en el cual no se realizar pruebas, entre el 60 % y el 80% de todo el esfuerzo se ocupa en mantenimiento, dejando el otro 20 % en desarrollo, mientras que un software al cual se le aplica un 5 % de esfuerzo en pruebas logra disminuir el porcentaje de mantenimiento reduciendo este último a 50 %, es decir se logra eliminar un 30 % en el costo de manutención [JACOBSON, I, 2003].

En muchas de las organizaciones y empresas, los costos de las pruebas en el ciclo de desarrollo de software oscilan entre 30 % y 50 % del costo total de software que

supuestamente está aprobados e incluso ya están siendo comercializados. Esta situación es causada principalmente por no utilizar metodologías o modelos bien definidos y realizar pruebas de forma intuitiva lo que conlleva a no identificar un porcentaje elevado de deficiencias que puede tener el producto.

Es por todo esto que es necesario tener no solo un modelo o metodología bien definida para pruebas, sino también los procedimientos y características y herramientas necesarias que puedan detectar el mayor porcentaje de errores a la hora de la ejecución de estas.

En base a esto se propone una alternativa para contribuir al desarrollo y ejecución de pruebas para el software basándose en los requerimientos del cliente y los casos de prueba para poder lograr obtener el mayor porcentaje posible al testear y detectar los bugs existentes en los sistemas y aplicaciones.

1.2 Justificación

Hoy en día el uso y la cantidad de sistemas de software existentes se encuentran cada vez más involucrados en las actividades cotidianas del ser humano. Sin embargo, estos sistemas son responsables de varios problemas y desastres en los ámbitos donde son ocupados, ejemplo de ello son la perdida de información, filtración de datos personales, robo de identidades, entre otros que generan una desconfianza en el software.

Las pruebas nos ayudan a detectar errores para evitar todo tipo de desastres que puedan ocasionar. Es por ello que se pretende crear un modelo para la generación de pruebas como medio de verificación y validación de productos de software, con la finalidad de que las pruebas puedan generar la liberación de productos de alta calidad al mercado y sean sistemas funcionales que generen confianza a la hora de utilizar

dichos medios. Esto ayudará a la empresas a reducir tiempos, recursos humanos y generar mayor valor al producto generado.

1.3 Pregunta de investigación

¿Es posible la implementación de un modelo de pruebas como medio de verificación y validación de productos de calidad, que garantice la liberación de software de alta competitividad?

1.4 Objetivos

1.4.1 Objetivo general

Diseñar un modelo que ayude al área de pruebas en las empresas desarrolladoras de software en la generación y ejecución de medios de verificación y validación de productos de software de alta competitividad.

1.4.2 Objetivos específicos

- Analizar modelos existentes para la generación y ejecución de pruebas de software.
- Determinar la viabilidad para la creación de un nuevo modelo para la generación y ejecución de pruebas.
- Generar un modelo para la generación y ejecución de pruebas funcionales al software.

- Generar un modelo para la generación y ejecución de pruebas de carga al software.
- Generar un modelo para la generación y ejecución de pruebas integrales de software.
- Generar un modelo para la generación y ejecución de pruebas de seguridad al software.
- Generar un modelo para la generación y ejecución de pruebas de estrés al software.
- Realizar una comparación entre en modelo generado y el resto de los modelos para probar sistemas de software.

1.5 Alcances y limitaciones

La creación de este modelo para la generación y ejecución de pruebas como medio de verificación y validación de productos de software le servirá a todas aquellas empresas, grupos de trabajo u organizaciones relacionadas en el ámbito de desarrollo software que tienen la necesidad de producir sistemas o aplicaciones de alta calidad con la finalidad de satisfacer la necesidades cambiantes del cliente y el mercado en general.

En la presente investigación se realizará la creación de un modelo que apoye en las pruebas a los sistemas. Así mismo cabe mencionar que dicha investigación no pretende realizar una nueva norma o modelo de pruebas, sin embargo se hará una mejora al modelo de interacción. Dentro de las pruebas que involucraran dicho modelo serán, pruebas funcionales, pruebas integrales, pruebas de carga, pruebas de rendimiento y pruebas de seguridad. Dicho modelo también presentara la forma en cómo se ejecutaran y reportaran evidencias de dichas comprobaciones antes de poder liberar un sistema o producto.

Finalmente para validar dicho modelo se implementaran las pruebas a un sistema de software con la finalidad de verificar el grado de satisfacción y calidad que se ha producido después de la liberación del producto y que respalden y verifiquen el uso de este modelo.

1.6 Descripción del documento

El presente documento está estructurado en 4 capítulos; la correspondiente bibliografía y una sección de anexos.

El capítulo 1 contiene una breve introducción, la descripción del problema, justificación del trabajo, la pregunta de investigación que se plantea, los objetivos, alcances y limitaciones, así como el análisis del estado del arte de los temas que competen al desarrollo de la metodología.
Dentro del capítulo 2 contiene el marco teórico de la investigación realizada para la creación de la nueva metodología.
En el capítulo 3 se expondrá y describirá a detalle el modelo de pruebas propuesto, así como su proceso y herramientas necesarias para llevar a cabo este de forma satisfactoria
Posteriormente en el capítulo 4 se presentaran los proyectos y la forma en como dicha metodología les fue aplicada.
Finalmente dentro del capítulo 5 se tendrán los resultados arrogados de cada uno de los proyectos a los cuales les fue aplicada la metodología y su relación entre ellos, para determinar el grado de eficiencia y eficacia del modelo propuesto.

1.7 Análisis del estado del arte

En esta sección se presenta una breve descripción de algunas investigaciones realizadas en el área de Ingeniería de software y de forma específica en las pruebas de software, en donde se presentan algunas metodologías y el resultado obtenido posterior a ejecución y presentación de los modelos propuestos.

1.7.1 Metodologías de pruebas de software

1.7.1.1 Investigación para la selección de un modelo de pruebas de software basado en la teoría de pruebas de Yin-Yang

En esta investigación se presenta una forma innovadora de seleccionar el modelo de pruebas adecuado para alcanzar una cantidad razonable y el resultado óptimo de las iteraciones en los productos. Se describe a detalle el contenido básico del modelo Yin-Yang y su teoría sobre las pruebas del software, en los cuales clasifica los diferentes tipos de pruebas en estas dos vertientes, por una parte el Yin en las cuales intervienen todas aquellas pruebas de caja blanca y el Yang en las cuales están clasificadas todas aquellas pruebas de caja negra, en base a esto propone un modelo con la combinación de estas dos ramas para probar los componentes; de acuerdo a la investigación de los datos del análisis de productividad del software en USA, con la consideración general sobre el tiempo, actividades, los recursos y necesidades, generar un modelo de pruebas de software que se compone en la distribución de la cantidad de pruebas.

El propósito de las pruebas de software es llevar acabo las operaciones en condiciones preestablecidas, por lo tanto, conocer los errores y evaluar al mismo tiempo la calidad del software en todas las etapas del proceso de desarrollo del software, cubriendo el ciclo completo de las pruebas para obtener un producto final que cumpla con los requerimientos del sistema.

Como resultado obtenido de dicha investigación de obtuvieron porcentajes considerables en cuando el rendimiento y detección de bugs en el proceso de desarrollo en comparación con los métodos tradicionales. El resultado de la prueba arrojo información de cómo se ha optimizado el rendimiento total con un aumentó 4.44% en contra posición de los métodos tradicionales. Es por ello que dicho método se considera es racional y eficaz, especialmente cuando se prueba aplicaciones a pequeña escala (Fangchun, 2012).

1.7.1.2 Pruebas de software adaptativo en el contexto de mejora controlada de un modelo de cadena de Markov

Las cadenas de Markov de forma general hacen referencia a procesos estocásticos en donde el resultado de una actividad previa es fundamental y afecta a la actividad siguiente. En el artículo habla sobre las pruebas de software de adaptación haciendo referencia a su contraparte con el control adaptativo en las pruebas de software. Esto significa que la estrategia de las pruebas de software debe ajustarse en base a los datos de prueba obtenidos durante pruebas de software en un entorno de mejora. Estudios previos sobre las pruebas de adaptación se basan en una cadena de Markov simplificada controlada (CMC) para el modelo de pruebas de software que emplea a varios supuestos poco realistas, es decir la información a utilizar en su mayoría es de escenarios casi imposibles de suceder y que a pesar de eso tienen que ejecutarse.

En este trabajo se propone un nuevo enfoque de adaptación de pruebas de software en el contexto de un modelo mejorado de la cadena de Markov (CMC) que tiene como objetivo eliminar este tipo de información irrelevante para las pruebas para general uno con mayor utilidad en los menores casos de prueba posibles. Un nuevo conjunto de supuestos sobre el proceso de pruebas de software con tipo de información más comunes en las pruebas reales de vida del software, tomando en cuenta que al inicio el software tendrá un determinado número de errores que bien podrían no ser

detectables y posterior a las pruebas encontrar la mayor cantidad de los posiblemente existentes, así como la depuración de bugs para eliminar aquellos que serán detectados en determinadas etapas. Finalmente el presente trabajo presento una mejora tanto en la cantidad de bugs encontrados como en el tiempo utilizado para la ejecución de las pruebas en comparación las metodologías tradicionales (Hu, 2008).

1.7.1.3 Método para generar casos de prueba funcional en el desarrollo de software

La calidad del software está determinada por una serie de factores involucrados en las pruebas del sistema, entre ellas las pruebas funcionales a partir de casos de uso del sistema, este artículo define un nuevo método para la generación de estas.

Para poder general los casos de prueba funcional utilizan de datos base los casos de uso y de los cuales se basa en las plantillas generadas y diagramas para generar cada escenario de prueba. Una vez realizado esto implementa un CheckList para comprobar que todos los componentes fueron probados. Este proceso consta de seis pasos, el primer es la identificación de los escenarios que serán probados, posteriormente validar que no existen escenarios repetidos, como tercer paso utilizar las plantillas previamente generadas para poder integrar la información, el seguida se ejecuta el proceso y finalmente verifica si existe algún otro escenario nuevo volviendo a pasos anteriores. Para la ejecución de esta actividad presenta una plantilla bien determinada para su apoyo.

Finalmente presenta una serie de resultados los cuales indican a su propuesta como una metodología viable para la generación de pruebas funcionales a partir de casos de uso, basado en el número de pasos que utiliza, utiliza menor esfuerzo y un punto muy importante diseña plantillas para su utilización (González, 2009).

1.7.1.4 Proceso de prueba para productos de software en un laboratorio de calidad

La calidad es uno de los factores de mayor importancia en las empresas desarrolladoras de software, debido a la naturaleza que los proyectos implican para sus usuarios. Es por ello que este autor presenta una propuesta basada en un proceso de pruebas de software para un laboratorio de calidad dentro de un ambiente universitario, así como los niveles de pruebas que se aplican a varios casos de estudio para su comprobación con ayuda de herramientas que automatizan algunas técnicas específicas.

Este estudio se realizó en la institución “José Antonio Echeverría” debido al mal proceso de pruebas que se tenía, o bien la forma en que se desarrollaban, es decir que no se daba la importancia necesaria a pesar de tener implantado un modelo de calidad como lo es CMMI el cual establecía cuatro etapas; la identificación, elaboración, ejecución y transferencia. Si bien tenían esta certificación en la parte de pruebas necesitaban implementar un proceso bien estructurado que cumpliera con lo especificado para que todo el software que se generara en dicha institución fuera de calidad y cumpliera con los requerimientos necesarios.

La propuesta definida por el autor consiste en cuatro niveles; pruebas iniciales correspondientes a la planeación y pruebas de código, la segunda son las pruebas del sistema, donde se enfoca a la validación de pruebas de rendimiento, de seguridad y funcionales, en la tercer etapa están las pruebas de aceptación, que involucra la verificación de pruebas de rendimiento y seguridad, intervención con el cliente y las pruebas alfa, y finalmente están las pruebas de explotación que involucran la implantación en un entorno del cliente con la versión del software Beta. Así mismo propone una seria de roles y responsabilidades para el proceso, entre los cuales se encuentra en Líder de laboratorio de pruebas, el Especialista de pruebas, el Representante del proyecto y el Probador.

De igual forma para poder cumplir con el proceso completo, define un flujo de trabajo para el proceso de pruebas y sus respectivos actores con sus responsabilidades que van desde la solicitud hasta su ejecución y conclusión.

Finalmente se obtuvieron resultados considerables en todos los proyectos, ya que no contaban con un proceso bien definido para las pruebas, con ello elevaron en un 100 % sus indicadores en la totalidad de los proyectos de ejecución (Justiz, 2014).

1.7.1.5 Generación de casos de prueba a partir de casos de uso en las pruebas de software

En este artículo el autor plasma la importancia de los casos de uso como una herramienta fundamental para detallar la funcionalidad de un sistema de software así como la forma en que estos han sido adoptados por la mayoría de quienes se dedican a desarrollar software y a su vez como estos son la base para poder general los casos de pruebas a ejecutar al software para obtener un producto de calidad, incluso Ivar Jacobson indica que los casos de uso forman parte base de muchos procesos en la ingeniería de software.

Es esta investigación menciona dos tipos de pruebas que pueden ser ejecutadas, las basadas en casos de uso mediante caja negra en donde se ven las funcionalidades del sistema en su forma externa y las basadas en las realizaciones de los casos de uso del diseño que son los correspondientes a módulos integrados y se prueba el sistema como caja blanca para validar el funcionamiento interno de sus componentes. En base a esto como se puede ver el autor menciona que los casos de uso determinan aspectos como, ¿El cliente que obtendrá como producto final?, ¿Al programador que y como tendrá que desarrollar cada módulo para que funcione según lo especificado?, ¿Al documentador que partes son indispensables redactar? y ¿Al individuo que tiene que probar?

Los casos de uso tienen una estructura definida por flujos principales los cuales indican el procedimiento común y flujos alternos los cuales describen alternativas que podrían darse si el común no se ejecutara, estos a su vez se convierten el escenarios de pruebas a ejecutar para validar si su funcionamiento cumple con lo descrito.

A su vez los casos de prueba se componen de un nombre, una descripción, datos de entrada, condiciones y resultados esperados. El autor Jacobs menciona cuatro pasos fundamentales para la generación de casos de prueba mientras que Heumann solo tres. Una vez que se identifican los escenarios posibles en base a los casos de uso se generan estos mediante una matriz que indica el nombren del escenario, el flujo principal y alternos, y los valores a utilizar para la ejecución de estos.

Finalmente el literato menciona la importancia de los casos de uso para generar casos de prueba y a su vez su limitación de estos ya que no siempre son la mejor opción cuando se trata de probar requerimientos no funcionales los cuales no están contemplados para la generación de escenarios de prueba y quedan fuera del alcance. Sin embargo la creación de escenarios a partir de casos de uso provoca tener productos confiables, entregas en tiempo, forma y la eficiencia del sistema (Jordán, 2006).

1.7.2 Pruebas en software embebido

1.7.2.1 Modelo de proceso para pruebas de software embebido

Las características de un software embebido son en gran medida muy diferentes a las del software convencional debido a que este es integrado en un segundo sistema. Este artículo analiza dichas características, así mismo menciona un modelo mediante el cual realiza la comparación con su método para la ejecución, validación y verificación de pruebas de software embebido.

En la comparación presenta de forma congruente la deficiencia que presenta el modelo en V para ejecución de pruebas, entre las cuales está el no encontrar los errores en etapas tempranas y el no probar el sistema en el ambiente donde será instalado si no en un ambiente por decirlo así, local.

Como resultado del estudio del problema presenta un modelo en el cual logra detectar errores desde la fase de análisis de requerimientos y diseño de la arquitectura además de que dicha metodología es implementada en su mayoría en el ambiente final donde correrá o será utilizado el software lo cual reduce en gran medida factores como costo y tiempo. Para la ejecución de las pruebas se apoya de casos de prueba y herramientas para automatizar pruebas al código como, codeTEST y testbed, con ello además de lo mencionado logra detectar errores como la redundancia y el manejo de excepciones en el código que favorecen que la funcionalidad del sistema y el rendimiento sea el correcto mediante la monitorización del sistema para evitar la pérdida o exceso en el uso de memoria (Qian, 2012).

1.7.2.2 La efectividad de pruebas en tiempo real de software embebido

El software embebido como se sabe, no es precisamente unos de los tipos de software fáciles de probar debido a su complejidad y estructura que estos guardan, además se der sistemas complejos para fines complejos y que requieren de un riguroso proceso de pruebas para poder ser liberados y que cumpla con las especificaciones del cliente. El presente autor expone una investigación argumentando la forma para ejecutar pruebas en tiempo real sobre software embebido.

Algunas de las características que el software embebido presenta son: una fuerte o rigurosa especificación que proporciona funciones muy específicas, características incrustadas o complementarias, características o procesos en tiempo real que hacen que las interacciones sean respuestas en tiempo y forma, finalmente la variedad al realizar software muy por encima del software convencional orientado a objetos o

estructurado lo que lleva a una forma diferente de probarlos. Todo esto conlleva a tener que analizar bien como está conformado para poder general las pruebas necesarias y correctas, por ejemplo; este tipo de software tiene características como, una fuerte persistencia lo que provoca buscar herramientas adicionales para probar algunos artefactos en particular, los casos de prueba suelen ser muy extensos y numerosos con situaciones complejas y el tener un ambiente de desarrollo muy diferente o no igual a donde se instalara, lo que provoca tener que probar en numerosas ocasiones un módulo antes de su integración en el sistema final.

Para este tipo de pruebas se toman en cuenta objetivos como la fiabilidad y la seguridad, así como la capacidad y la eficacia presentando una fórmula para la cobertura de las pruebas que es la siguiente:

$$CR = \left(\frac{ie}{it}\right) x100\ \%$$

En donde:
CR= tasa de cobertura
ie= número de elementos ejecutados al menos una vez
it = número total de elementos

Finalmente todo lo anterior lo aplica a un caso de estudio en donde obtiene resultados considerables en las pruebas y revisiones encontrando errores con precisión y rapidez con lo que el autor da a conocer resultados sobre pruebas en tiempo real mediante un método para la cobertura completa de las pruebas, además de dar a conocer o referenciar la alta fiabilidad en estos procesos (Zhang, 2011).

1.7.3 Pruebas de seguridad

1.7.3.1 Software de pruebas de seguridad

Las pruebas de seguridad se han mudado recientemente más allá del ámbito del puerto de escaneado en red para incluir comportamiento de prueba del software como un aspecto crítico del comportamiento del sistema. Desafortunadamente, las pruebas de software de seguridad son una tarea común mal entendida.

Una prueba realizada correctamente, es algo más profundo que una simple caja negra y un sondeo en la capa de presentación (el tipo realizada por denominadas herramientas de seguridad de aplicaciones) e incluso más allá de la prueba de funcionamiento del aparato de seguridad.

Los probadores deben usar enfoques basados en el riesgo, basadas tanto en la realidad arquitectónica del sistema y la mentalidad del atacante, para medir el software de seguridad adecuado. Mediante la identificación de riesgos en el sistema y la creación de pruebas conducidas por esos riesgos, un probador de software de seguridad correctamente puede centrarse en las áreas de código en el que un ataque es probable que tenga éxito. Este enfoque proporciona un mayor nivel de seguridad del software de seguridad que es posible con las pruebas de caja negra clásicas (Potter, 2010).

1.7.3.2 Pruebas de software de seguridad basado en SSD típico: Un caso de estudio

Debido a la creciente complejidad de las aplicaciones Web tradicionales y que sólo se ponen a prueba y validan los mecanismos de seguridad de software, cada vez son más ineficaces para detectar defectos latentes de seguridad (SSD). El número de denuncias de vulnerabilidades de aplicaciones web está aumentando dramáticamente. Sin embargo, la mayor parte de las vulnerabilidades son resultado de algún SSD típico.

Este trabajo presenta una prueba del software de seguridad eficaz (SST), el cual se extiende el proceso tradicional de pruebas de seguridad para el análisis del comportamiento defectos que incorpora ventajas del método de análisis tradicional y SSD es una metodología basada en pruebas de seguridad. Las principales aplicaciones muestran la eficacia del modelo de prueba (Hui, 2010).

1.7.4 Aportes en general de pruebas de software

1.7.4.1 Aportes y perspectivas en arquitecturas de entornos de pruebas de software

La producción de sistemas de software de alta calidad ha sido una de las preocupaciones más importantes del desarrollo de software. En esta perspectiva, la Arquitectura de Software y Testing de Software son dos áreas importantes de investigación que han contribuido en esa dirección.

La atención prestada a la arquitectura de software ha desempeñado un papel significativo en la determinación del éxito de los sistemas de software. De lo contrario, las pruebas de software ha sido reconocida como una actividad fundamental para asegurar la calidad del software, sin embargo, es un costoso y es propenso a errores, y es una actividad que consume tiempo. Por esta razón, una diversidad de herramientas de prueba y entornos se ha desarrollado, sin embargo, han sido casi siempre diseñados sin una adecuada atención a su evolución, el mantenimiento, la reutilización, y sobre todo a sus arquitecturas. Por lo tanto, este documento presenta los principales aportes de sistematizar el desarrollo de herramientas de prueba y entornos, con el objetivo de mejorar su calidad, la reutilización y la productividad.

En particular, nos hemos ocupado de las arquitecturas para las herramientas de prueba de software y entornos, y también han desarrollado y puesto a herramientas de análisis disponibles. Asimismo, las perspectivas de estado de la investigación en

esta área, incluyendo los temas de investigación abiertas que deben ser tratadas, considerando la relevancia incuestionable de las pruebas de automatización para la actividad de prueba (Nakagawa, 2011).

1.7.4.2 Una encuesta Industrial en los aspectos contemporáneos de pruebas de software

Las pruebas de software son una fuente importante de gastos en proyectos de software y un proceso de prueba adecuado es un ingrediente esencial en el desarrollo rentable de software de alta calidad. Aspectos contemporáneos, tales como la introducción de un proceso más ligero, las tendencias hacia el desarrollo distribuido, y el rápido aumento de software en sistemas embebidos y seguridad críticos, desafían el proceso de pruebas de maneras inesperadas.

A nuestro entender, hay muy pocos estudios que se centran en estos aspectos en relación a las pruebas según la percepción de diferentes colaboradores en el proceso de desarrollo de software. En este trabajo se analiza cualitativamente y cuantitativamente los datos de una encuesta industrial, con especial atención a las prácticas actuales y las preferencias sobre aspectos contemporáneos de pruebas de software.

En concreto, el análisis se centra en las percepciones del proceso de pruebas de software en las diferentes categorías de encuestados. Categorización de los encuestados se basa en la seguridad-criticidad, la agilidad, la distribución del desarrollo y el dominio de la aplicación. Además de confirmar algunos de los hechos comúnmente reconocidos, los resultados también revelan diferencias notables entre las prácticas de ensayos preferidos y los reales. Creen que los esfuerzos continuos de investigación son esenciales para proporcionar directrices para la adaptación del

proceso de pruebas para cuidar de estas discrepancias, lo que mejora la calidad y la eficiencia del desarrollo de software (Causevic, 2010).

1.7.4.3 El estudio sobre un inteligente uso general de la suite de pruebas de software automatizado

Para realizar un intenso trabajo de pruebas de software automatizado, se presenta el diseño e implementación de un inteligente sistema de uso general de una suite de pruebas automatizadas de software. Con las dos herramientas principales en la suite: un sistema automatizado de pruebas de software planificador, Arnés, y un analizador de archivo de registro de errores, *Logscanner*, el informe de síntesis se puede enviar al ingeniero de pruebas inmediatamente después de que se termine la prueba a través de correo electrónico, incluso cuando el ingeniero de pruebas está lejos del lugar de la prueba.

La funcionalidad y las características de la suite de propósito general de pruebas de software automatizado ayudar a aumentar la eficiencia de las pruebas, reduce la intensidad de trabajo de pruebas de software, que a su vez, conduce a la disminución en la prueba de costo (Wu, 2010).

1.7.4.4 La investigación y aplicación de métodos de gestión del conocimiento en proceso de pruebas de software

En las organizaciones de pruebas de software, la gestión del conocimiento eficaz del proceso de pruebas es la clave para mejorar la calidad de las pruebas de software. Una de las cuestiones más importantes es cómo integrar eficazmente a la gestión del conocimiento en el proceso de pruebas de software para que los activos de conocimiento se puedan transmitir y ser reutilizados en organizaciones de análisis de software.

En este trabajo, el estado actual de la gestión del conocimiento en las pruebas de software se analizó en la problemática existente y se identificaron los principales

objetivos en un sistema de gestión del conocimiento en las pruebas de software que se diseñó e implementó. Por el momento, muchos de ellos basados en ontologías tecnologías clave se han propuesto, que podría ser utilizado en la representación del conocimiento de pruebas de software, búsqueda y clasificación del documento de conocimiento, y la construcción del mapa de conocimiento. Finalmente, un ejemplo del módulo de mapa de conocimiento fue dado a verificar que la gestión del conocimiento en las pruebas de software es razonable y eficaz (Xue-Mei, 2009).

Capítulo 2 Marco teórico

Este capítulo contiene el fundamento teórico, conceptos necesarios y apropiados que permitirán aclarar el resto del proceso llevado a cabo en el presente trabajo. Tal soporte incluye información sobre los diferentes tipos de pruebas y el aseguramiento de la calidad del software. Esta sección también contiene información descriptiva de algunos modelos y el proceso que tiene cada uno y como se involucra en las área de pruebas en dicha metodología.

2.1 Calidad

El hablar de calidad involucra factores pertenecientes al grado de satisfacción del cliente, anteriormente el software que era desarrollado y posteriormente entregado a los usuarios finales tenía problemas como la fiabilidad, mantenibilidad, reutilización, seguridad, usabilidad, entre otros. Estos problemas llevaron al desarrollo e implementación de métodos y modelos formales para la gestión de calidad del software, con la finalidad de tratar de reducir en lo posible factores como los mencionados.

Pala lograr un software con alta calidad se deben llevar a cabo 4 pasos importantes como son: usar prácticas y procesos de ingeniería del software que hayan sido probados, una buena administración del proyecto, un control de calidad exhaustivo y contar con la infraestructura de aseguramiento la calidad.

Dentro de la ingeniería del software varios autores han tratado de definir el concepto de calidad del software y están de acuerdo con la importancia que tiene para obtener

aplicaciones que cumplan con todos los requisitos establecidos y además tengan un rendimiento óptimo para el cliente.

David Garving (1984), dice que “la calidad es un concepto complejo y de facetas múltiples” y puede describirse desde 5 puntos de vista diferentes. El punto de vista transcendental dice que la calidad es algo que se reconoce inmediatamente, pero no se puede definir explícitamente. El punto de vista del usuario concibe la calidad en términos de las metas específicas del usuario final y si el producto las satisface tiene calidad. El punto de vista del fabricante la define en términos de las especificaciones originales del producto. Si éste las cumple tiene calidad. El punto de vista del producto sugiere que la calidad tiene que ver con las características inherentes de un producto. El punto de vista basado en el valor la mide de acuerdo con lo que un cliente éste dispuesto a pagar por un producto (Garving, 84).

Por otra parte Robert Glass (1998), realiza una pregunta correspondiente a la calidad del producto. ¿Son la calidad del diseño y de la conformidad los únicos aspectos que se deben considerar? Afirma que es mejor plantear una relación más intuitiva definida en la siguiente formula:

Satisfacción del usuario = producto que funcione + la buena calidad + entrega dentro del presupuesto y plazo

Glass afirma que la calidad es importante, pero que si el usuario está insatisfecho nada de lo demás importa (Glass, 1998).

Una definición más concreta propuesta por Roger S. Pressman, sugiere que “la calidad de software es el cumplimiento de los requisitos de funcionalidad y desempeño explícitamente establecidos, de los estándares de desarrollo explícitamente documentados y de las características implícitas que se esperan de todo software desarrollado profesionalmente” (Pressman, 2010).

2.1.1 Factores que determinan la calidad de McCall

En el año de 1977, McCall propuso una clasificación útil de los factores de calidad que afectan la calidad del software y que hasta en la actualidad siguen vigentes.

McCall planteó una clasificación basada en tres aspectos importantes del producto de software: características de operación, que incluye corrección, confiabilidad, usabilidad, integridad y eficiencia; transición, compuesta por portabilidad, reutilización y compatibilidad; y por último revisión de un producto, donde se encuentran factores como facilidad de mantenimiento, flexibilidad y facilidad de prueba (McCall, 1977).

2.1.2 Factores de la calidad ISO 9126

El estándar ISO 9126 se desarrolló con la intención de identificar los atributos clave del software e identifica seis atributos clave de la calidad.

1. Funcionalidad. Es el grado en el que el software satisface las necesidades de: adaptabilidad, exactitud, interoperabilidad, cumplimiento y seguridad.
2. Confiablidad. Es la cantidad de tiempo que el software se encuentra disponible para uso según los siguientes atributos: madurez, tolerancia a fallas y recuperación.
3. Usabilidad. Grado en el que el software es fácil de usar, que sea entendible y operable.
4. Eficiencia. Empleo de los recursos del sistema óptimamente por parte del software, como el comportamiento del tiempo y de los recursos.
5. Facilidad de recibir mantenimiento. Facilidad con que se llevan a cabo las reparaciones del software: que sea analizable, cambiable, estable y susceptible someterse a pruebas.

6. Portabilidad. Facilidad con que el software puede llevarse de un ambiente a otro: debe ser adaptable, instalable, conformidad y sustituible.

2.1.3 El costo de la calidad.

El lograr la calidad del software cuesta tiempo y dinero. El costo de la calidad incluye todos los costos en los que se incurre al buscar la calidad o al realizar actividades relacionadas con ella y los costos posteriores por la falta de calidad.

Los costos de prevención incluyen:

- El costo de actividades de administración requeridas para planear y coordinar todas las actividades de control y aseguramiento de la calidad.
- El costo de las actividades técnicas agregadas para desarrollar modelos completos de los requerimientos y del diseño.
- Los costos de planear las pruebas.
- El costo de toda la capacitación asociada con estas actividades.

Los costos de a evaluación incluyen las actividades de investigación de la condición del producto la “primera vez” que pasa por cada proceso. Los costos de falla son aquellos que se eliminan si no hubiera errores antes o después de enviar el producto a los consumidores y se dividen en internos y externos. Se incurre en costos internos de falla cuando se detecta un error en el producto antes del envío. Los costos externos de falla son los que se encuentra después de que el producto se envía los consumidores.

En la industria del software se gasta alrededor del $0.50 por cada $1.00 gastado en el desarrollo y mantenimiento a la hora de encontrar y corregir errores. La mayoría de los métodos están por debajo del 35% en la eficiencia de la remoción de defectos o bien lograr eliminar tan solo 1 de cada 3 errores existentes. Todas las pruebas juntas rara vez son superiores al 85% de eficiencia en la remoción de bugs. Aproximadamente el

7% de las correcciones de errores incluyen nuevos bugs. Alrededor del 6% de los casos de prueba tienen errores propios (Jones, 2012).

2.1.4 Control de la calidad

El control de la calidad consta de un conjunto de acciones que ayudan a asegurar que el producto cumpla sus metas de calidad. Los modelos revisan que estén completos y que son consistentes. El código es inspeccionado para encontrar y corregir errores antes que comiencen las pruebas. Son aplicadas un una serie de etapas de prueba para detectar los errores de procesamiento lógico, manipulación de datos y comunicación con la interfaz.

2.1.5 Aseguramiento de la calidad

El aseguramiento de la calidad establece la infraestructura de apoyo a los métodos de la ingeniería del software, la administración de proyectos y las acciones que se llevan a cabo en el control de la calidad. El aseguramiento de la calidad consiste en un conjunto de funciones de auditoría y reportes para evaluar la eficacia y completitud de las acciones de control de la calidad. Su meta es proveer al equipo de administración y técnico los datos necesarios para mantener informado sobre la calidad de producto y obtener perspectiva y confianza en que el producto funcionara correctamente.

La primera función formal de aseguramiento y control de la calidad se introdujo en los laboratorios Bell en 1916, actualmente todas las compañía tienen mecanismos para asegurar la calidad en sus productos.

2.1.5.1 Elementos de aseguramiento de la calidad del software

El aseguramiento de la calidad del software incluye un rango amplio de preocupaciones y actividades que se centran en la administración de la calidad del software.

Pruebas. Las pruebas de software tienen como objetivo principal detectar errores.

Revisiones y auditorias. Las revisiones técnicas tienen como objetivo es detectar errores. Las auditorias se realizan con la intención de garantizar que se siga los lineamientos de calidad en el trabajo de la ingeniería del software.

Estándares. Existen distintas organizaciones que establecen estándares y han producido una amplia variedad de ellos para ingeniería del software y documentos relacionados.

Seguridad. El software es un componente crucial de los sistemas humanos y la consecuencia de los defectos ocultos puede causar una catástrofe.

Administración de riesgos. Se debe garantizar que las administraciones de riesgos se efectúen de manera apropiada y que se establezcan planes de contingencia relacionados con los riesgos.

2.2 Definiciones y estrategias de prueba de software

Una estrategia de prueba de software proporciona una metodología que describe los pasos que deben realizarse como parte de una prueba, cuando se planean y se llevan a cabo dichos pasos, y cuanto esfuerzo, tiempo y recursos se requieren.

La prueba en un conjunto de actividades que pueden planearse por adelantado y realizarse de manera sistemática. Durante el proceso de software, debe definirse una

plantilla para la prueba de software que es un conjunto de pasos que incluyen métodos de prueba y técnicas de diseño de casos de prueba específicos.

2.2.1 Definiciones para pruebas de software

Según Sommerville el objetivo de las pruebas es demostrar que un programa hace lo que se definió por el cliente, así como descubrir defectos en el programa antes de usarlo. Al probarse el software, se ejecutara con información que en teoría el software podría manejar sin inconvenientes, para con ello detectar posibles anomalías en el sistema o información de atributos no funcionales del programa.

También menciona que el proceso de pruebas tiene dos metas distintas:

1. Por una parte demostrar al proveedor y cliente que el software cumple con los requerimientos definidos en las etapas iniciales. Para ello debe de haber al menos una prueba o más por cada requerimiento con diversas combinaciones que se podrían esperar por el usuario.
2. Además de encontrar situaciones donde el comportamiento del software es incorrecto, indeseable o no esté de acuerdo con sus especificaciones. Estas son provocadas por defectos en el sistema y que necesitan ser reparadas antes de ser entregado con el usuario final.

Así mismo es importante presentar algunas definiciones donde podría haber un poco de confusión por lo cual es importante aclarar los siguientes términos:

- Prueba: Una actividad en la que un sistema o componente es ejecutado bajo condiciones especificadas , los resultados son observados o registrados y una evaluación es hecha de algún aspecto del sistema o componente (IEEE, 1990]).

- Caso de prueba: Es un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y poscondiciones de ejecución, desarrollados con un objetivo particular o condición de prueba, tal como ejercitar un camino de un programa particular o para verificar que se cumple un requerimiento especifico (IEEE, 1990).
- Procedimiento de prueba: Instrucciones detalladas para la configuración, ejecución y evaluación de los resultados para un caso de prueba determinado (IEEE, 1990).

- Resultado real: Es el comportamiento producido u observado cuando un componente o sistema es probado (International Software Testing Qualifications Board, 2005).

- Resultado esperado: Es el comportamiento predicho por la especificación u otra fuente, del componente o sistema a ser probado bajo condiciones especificadas (International Software Testing Qualifications Board, 2005).

- Programa de prueba: Es un programa que específica para la prueba: elemento a ser probado, requerimiento, estado inicial, entradas, resultados esperados y criterio de validación (Beizer, 1990).

- Conjunto de prueba: Es un conjunto de una o más pruebas, con un propósito y base de datos común que usualmente se ejecutan en conjunto (International Software Testing Qualifications Board, 2005).

- Ciclo de prueba: Es la ejecución del proceso del Testing contra una versión identificada del producto a probar (International Software Testing Qualifications Board, 2005).

- Datos de prueba: Son los datos que existen (por ejemplo, en una base de datos) antes de que una prueba sea ejecutada, y que afecta o es afectado por el componente o sistema a probar (International Software Testing Qualifications Board, 2005).
- Ejecución de prueba: Es el proceso de ejecutar una prueba para el componente o sistema a probar, produciendo el resultado real (International Software Testing Qualifications Board, 2005).

- Incidente: cuando una falla sucede, puede o no ser evidente para el usuario (cliente o probador). Un incidente es el síntoma, asociado con la falla, que alerta al usuario de la ocurrencia de una falla (Jorgensen, 2002).

- Pruebas de caja negra: Este tipo de pruebas es también conocido como pruebas funcionales o pruebas de comportamiento, se centran en obtener conjuntos de condiciones de entrada que ejerciten completamente los requisitos funcionales de un programa (Pressman, 2010).

El objetivo principal de las pruebas de caja negra es encontrar ítems de las siguientes categorías (Pressman, 2010):

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en acceso a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

- Pruebas de caja blanca: En este enfoque de pruebas, llamado también pruebas estructurales, los casos de prueba se obtienen de la estructura interna del código del software bajo prueba y la prueba pasa sólo si los resulta-dos son correctos (Pressman, 2010).

- Pruebas de camino: Las pruebas de caminos son una estrategia de pruebas estructurales cuyo objetivo es bar cada camino de ejecución independiente en un componente o programa (Sommerville, 2005).
- Pruebas de condición: La prueba de condición es un método de prueba de caja blanca en la que se ejercitan las condiciones lógicas contenidas en el módulo de un programa (Pressman, 2010).

- Pruebas unitarias: Las pruebas unitarias, también llamadas pruebas de componentes, se encargan de probar, individualmente, subprogramas, subrutinas o procedimientos en un programa (Myers, 1979).

- Pruebas funcionales: En general, los componentes de software son componentes compuestos constituidos por varios objetos en interacción. Por consiguiente, la prueba de componentes compuestos tiene que enfocarse en mostrar que la interfaz de componente se comporta según su especificación (Sommerville, 2011).

- Pruebas integrales: Las pruebas de integración son una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar los componentes probados de manera individual y construir una estructura de programa que se haya dictado por diseño (Pressman, 2010).

- Pruebas de integración descendente: La prueba de integración descendente es un enfoque incremental a la construcción de la arquitectura de software. Los módulos se integran al moverse hacia abajo a través de la jerarquía de control, comenzando con el módulo de control principal (programa principal). Los módulos subordinados al módulo de control principal se incorporan en la estructura en una forma de primero en profundidad o primero en anchura (Pressman, 2010).

- Pruebas de integración ascendente: La prueba de integración ascendente, como su nombre implica, comienza la construcción y la prueba con módulos atómicos (es decir, componentes en los niveles inferiores dentro de la estructura del programa). Puesto que los componentes se integran de abajo hacia arriba, la funcionalidad que proporcionan los componentes subordinados en determinado nivel siempre está disponible y se elimina la necesidad de representantes (Pressman, 2010).

- Pruebas de regresión: Es la nueva ejecución de algún subconjunto de pruebas que ya se realizaron a fin de asegurar que los cambios no propagaron efectos colaterales no deseados (Pressman, 2010).

- Pruebas Alfa: Se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El software se usa en un escenario natural con el desarrollador “mirando sobre el hombro” de los usuarios y registrando los errores y problemas de uso. Las pruebas alfa se realizan en un ambiente controlado (Pressman, 2010).

- Pruebas Beta: La prueba beta se realiza en uno o más sitios del usuario final. A diferencia de la prueba alfa, por lo general el desarrollador no está presente. Por tanto, la prueba beta es una aplicación “en vivo” del software en un ambiente que no puede controlar el desarrollador (Pressman, 2010).

- Pruebas de seguridad: La pruebas de seguridad intenta verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier penetración impropia (Pressman, 2010).

- Pruebas de esfuerzo: Las pruebas de esfuerzo se ejecuta un sistema en forma que demanda recursos en cantidad, frecuencia o volumen anormales (Pressman, 2010).

- Pruebas de despliegue: En muchos casos, el software debe ejecutarse en varias plataformas y bajo más de un entorno de sistema operativo. La prueba de despliegue, en ocasiones llamada prueba de configuración, ejercita el software en cada entorno en el que debe operar (Pressman, 2010).

- Pruebas de sistema: Se refiere al comportamiento del sistema entero. La mayoría de las faltas funcionales deben haber sido identificadas ya durante las pruebas de unidad e integración. La prueba del sistema generalmente se considera apropiada para probar los requerimientos no funcionales del sistema, tales como seguridad, desempeño, exactitud, y confiabilidad. Las interfaces externas, los dispositivos de hardware, o el ambiente de funcionamiento también se evalúan a este nivel (IEEE, 2004).

Dentro de las pruebas funcionales como se mencionaban, existe una gran variedad de ellas que a continuación se describirán (Myers, 2004):

- Prueba de volumen: Este tipo de prueba del sistema somete el programa a grandes volúmenes de datos. El propósito es demostrar que el programa no puede manejar el volumen de datos especificados en sus objetivos. El

volumen a probar puede requerir recursos significativos, en términos del tiempo de procesador y de personas.

- Pruebas de estrés: Estas pruebas someten al programa a cargas pesadas. Una carga pesada es un volumen máximo de datos, o de actividad, en un plazo corto de tiempo. La prueba de estrés implica como elemento el tiempo.

- Pruebas de usabilidad: Intentan encontrar problemas de usabilidad del sistema en su interacción con humanos. El análisis de factores humanos es una cuestión altamente subjetiva.

- Pruebas de desempeño: Muchos programas tienen objetivos específicos de desempeño o de eficiencia, indicando características tales como tiempos de respuesta y rendimiento bajo ciertas condiciones de carga de trabajo y configuración. En la prueba de desempeño, los casos de prueba se deben diseñar para demostrar que el programa no satisface sus objetivos de desempeño.

- Pruebas de almacenamiento: Algunos programas tienen establecidos límites en el almacenamiento, que indican, por ejemplo, la cantidad de memoria principal y secundaria que el programa utiliza. Se diseñan los casos de prueba para demostrar que estos objetivos de almacenamiento no se han resuelto.

- Pruebas de configuración: Implican probar el programa con distintas configuraciones de hardware y software posibles.

- Pruebas de compatibilidad: La mayoría de los programas que se desarrollan no son totalmente nuevos; son a menudo reemplazos de un sistema existente. Algunas veces, estos programas tienen objetivos específicos referentes a su compatibilidad y procedimientos de conversión del sistema existente. Se diseñan los casos de prueba para demostrar que los objetivos de la compatibilidad no se han resuelto y que los procedimientos de conversión no funcionan.

- Pruebas de instalación: La prueba de instalación es una parte importante del proceso de prueba del sistema, particularmente en un sistema empaquetado con instalación automatizada.

- Pruebas de confiabilidad: Estos tipos de prueba tienen como objetivo la mejora de la confiabilidad del programa, probando que las declaraciones específicas sobre confiabilidad se cumplen.

- Pruebas de recuperabilidad: Algunos programas tienen requerimientos específicos de recuperación que indican cómo el sistema se recupera de errores de programación, de faltas del hardware, y de errores en los datos. Un objetivo de la prueba del sistema es demostrar que estas funciones de recuperación no funcionan correctamente.

- Pruebas de mantenimiento: Algunos programas tienen objetivos de mantenimiento, como por ejemplo: requerimientos específicos sobre las ayudas que se proporcionarán al sistema, los procedimientos de mantenimiento, y la calidad de la documentación de la lógica interna.

- Estrategia: Serie de acciones/actividades a realizar para establecer un marco de trabajo.

- Producto: Documento que es requerido en cada una de las fases y/o documentación con la que debe cumplir un rol determinado.

- Incidencia: Documento que es requerido en cada una de las fases y/o documentación con la que debe cumplir un rol determinado.

- Lecciones aprendidas: Son el conjunto de éxitos y errores que el equipo de trabajo ha logrado manejar durante el o los proyectos.

- Error: Se refiere a un error humano, una equivocación, i.e., algo que una persona piensa o hace y que no debería de hacer (Jorgensen, 2002).

- Defecto: Se la manifestación de un error. Esto puede expresarse en otra forma como, es el defecto es una representación de un error, una carencia o algo que está mal ya sea en los diagramas de flujo, en casos de uso, en el código fuente, etc. Cuando se comenten errores de omisión, el defecto resultante es algo que falta y que debería estar presente. Por lo tanto, como Jorgensen menciona, debemos hablar de defectos de comisión y defectos de omisión. Los primeros ocurren cuando se introduce algo en una representación que es incorrecta. Los defectos de omisión ocurren cuando no introducimos alguna información que es correcta. Como es de imaginarse, este último tipo de defectos es el más difícil de detectar y de resolver. El término bug, muy comúnmente usado en el ámbito de la computación, es usado para referirse a un defecto (Jorgensen, 2002).

- Falla: Es un comportamiento incorrecto del software debido a la activación de un defecto. Hay dos cosas que debemos remarcar aquí: la primera, las fallas sólo ocurren en una representación ejecutable del código fuente; la segunda, la definición de fallas, sólo corresponde a los defectos de comisión (Jorgensen, 2002).

A continuación se ejemplifica la relación y diferencia entre estos tres términos.

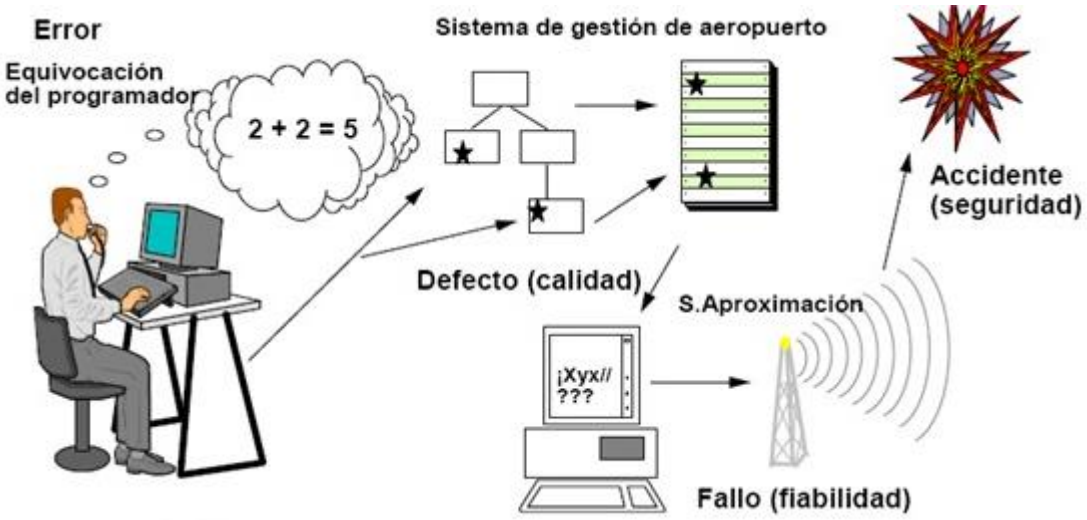


Figura 1. Diferencia entre error, defecto, falla e incidencia (Jorgensen et al., 2002).

- Dato de prueba: es un dato que ha sido seleccionado específicamente para ser usado en la prueba de un programa de computadora.

- Caso de prueba: un conjunto de entradas (datos) de prueba, condiciones de ejecución y resultados esperados desarrollados para un objetivo en particular, como ejecutar una ruta de programa en particular o verificar el cumplimiento de un requerimiento específicos (IEEE, 1990).

- Datos de entrada: es el conjunto de todos los posibles datos de entrada que puede recibir un programa. Esto incluye las variables globales, los parámetros recibidos por una función o las entradas que puedan ser introducidas externamente (IEEE, 1990).

- Requerimientos: Los requerimientos del software expresan las necesidades y las restricciones puestas en un producto de software que contribuyen a la solución de un cierto problema del mundo real. Una característica esencial de todos los requerimientos del software es que sean comprobables. Una

afirmación es verificable si se puede diseñar un experimento que demuestre o refute la verdad de la sentencia (Beizer, 1990).

2.2.1 Verificación y validación de software

La verificación se refiere al conjunto de tareas que garantiza que el software realiza correctamente una función específica. La validación es un conjunto diferente de tareas que aseguran que el software sigue los requerimientos del cliente.

Según el estándar definido por la IEEE 1059-1993 “Guía para los planes de verificación y validación e software”, define a las pruebas como:

> “Es el proceso de analizar un elemento de software para detectarlas diferencias entre las condiciones existentes y las requeridas (es decir, bugs), y evaluar las características del elemento de software” (IEEE, 1994).

Así mismo, el estándar IEEE 610.12-1990 “Glosario de términos de Ingeniería de Software”, define pruebas como:

> “Actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, los resultados son observados o grabados, y se realizan una evaluación de algún aspecto del sistema o componente” (IEEE, 1990).

CMMI define el propósito de la Verificación y la Validación de la siguiente manera (CMMI, 2002):

- El propósito de la Verificación es asegurar que los Productos Internos seleccionados cumplen con su especificación de requerimientos. Los métodos de verificación pueden ser, entre otros: inspecciones, revisiones

por pares, auditorias, recorridas, análisis, simulaciones, pruebas y demostraciones.

- El propósito de la Validación es demostrar que un producto o componente de producto cumple su uso previsto cuando es puesto en su ambiente previsto. Deben ser seleccionados los productos internos (por ejemplo: requerimientos, diseño, prototipos) que mejor indican cuán bien el producto y los productos internos deben satisfacer las necesidades del usuario.

Por otro lado Kit define la verificación y la validación de la siguiente manera (Kit, 1995):

- La Verificación es el proceso de evaluar, revisar, inspeccionar y hacer controles de escritorio de productos intermedios, tales como especificaciones de requerimientos, especificaciones de diseño y código.

- La validación es ver si en base a lo verificado se cumple con las expectativas del cliente.

2.3 Modelos para desarrollo de software y pruebas

En la ingeniería de software existe una gran variedad de modelos y metodologías que ayudan al desarrollo de sistemas y productos, depende de cada empresa u organización adoptar cada una de estas para poder guiar todo el proceso de inicio a fin. Así mismo cada modelo contiene el área correspondiente a pruebas que es la que nos interesa describir para dar a entender donde se estará aplicando la metodología que se propone y describe en el capítulo 3.

La ingeniería de software implica principios fundamentales que deben guiarse siempre de forma general. Incluyen actividades explícitas para el entendimiento del problema y la comunicación con el cliente, métodos definidos para representar un diseño,

mejores prácticas para la implementación de la solución y estrategias y tácticas sólidas para las pruebas. Si se siguen los principios básicos, esto resulta en productos de alta calidad.

Para conseguir el objetivo de construir productos de alta calidad dentro de la planificación, la ingeniería del software emplea una serie de prácticas para:

- Entender el problema
- Diseñar una solución
- Implementar la solución correctamente
- Probar la solución
- Gestionar las actividades anteriores para conseguir alta calidad

La ingeniería del software representa un proceso formal que incorpora una serie de métodos bien definidos para el análisis, diseño, implementación y pruebas del software y sistemas. Además, abarca una amplia colección de métodos y técnicas de gestión de proyectos para el aseguramiento de la calidad y la gestión de la configuración del software (Pressman, 2010).

2.3.1 Modelos generales de proceso

Según Pressman el proceso de software es una estructura para las actividades, acciones y tareas que se requieren a fin de construir software de alta calidad. Un proceso del software define el enfoque adoptado mientras se hace ingeniería sobre el software. Pero la ingeniería de software también incluye tecnologías que pueblan el proceso: métodos técnicos y herramientas automatizadas.

En base a lo anterior, presenta de forma gráfica los diversos flujos generales de proceso en la elaboración de un software de la siguiente manera (Pressman, 2010):

- Un flujo de proceso lineal ejecuta cada una de las cinco actividades estructurales en secuencia, comenzando por la comunicación y terminando con el despliegue (véase la figura 2).
- Un flujo de proceso iterativo repite una o más de las actividades antes de pasar a la siguiente (véase la figura 3).
- Un flujo de proceso evolutivo realiza las actividades en forma “circular”. A través de las cinco actividades, cada circuito lleva a una versión más completa del software (véase la fi-gura 4).
- Un flujo de proceso paralelo (véase la figura 5) ejecuta una o más actividades en paralelo con otras (por ejemplo, el modelado de un aspecto del software tal vez se ejecute en paralelo con la construcción de otro aspecto del software).

Flujo de procesos:

Figura 2. Flujo de proceso lineal (Elaboración propia).

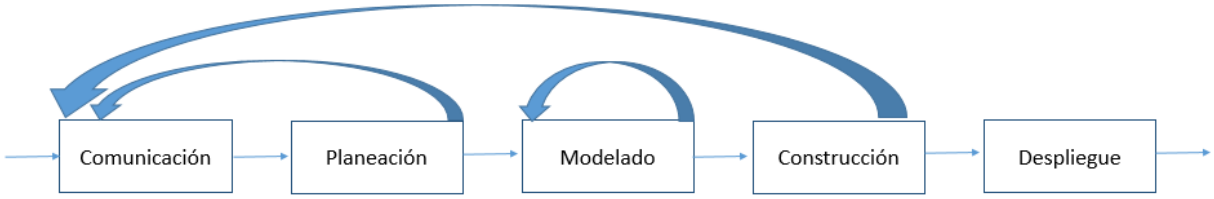


Figura 3. Flujo de proceso iterativo (Elaboración propia).

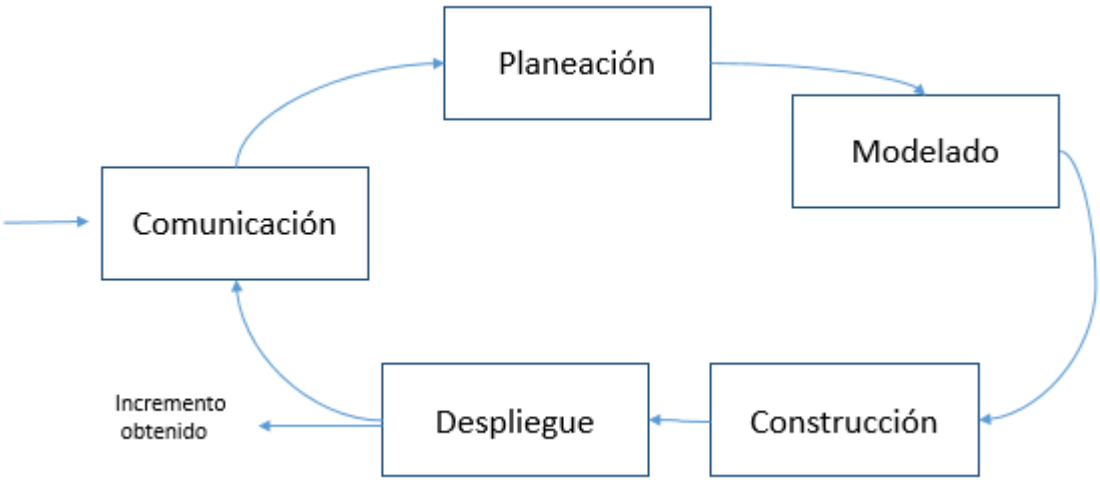


Figura 4. Flujo de proceso evolutivo (Elaboración propia).

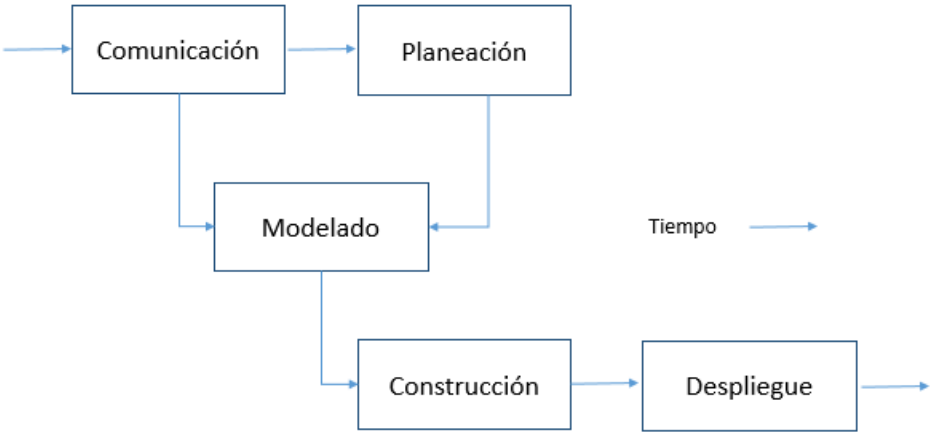


Figura 5. Flujo de proceso paralelo (Elaboración propia).

2.3.2 Modelo en cascada

Este tipo de modelo tiene una peculiaridad muy específica ya que se sigue un riguroso proceso lineal en donde para poder pasar a la siguiente fase debe estar completa la

anterior. Esta situación se encuentra en ocasiones cuando deben hacerse adaptaciones o mejoras bien definidas a un sistema ya existente (por ejemplo, una adaptación para software de contabilidad que es obligatorio hacer debido a cambios en las regulaciones gubernamentales). También ocurre en cierto número limitado de nuevos esfuerzos de desarrollo, pero sólo cuando los requerimientos están bien definidos y tienen una estabilidad razonable (Pressman, 2010).

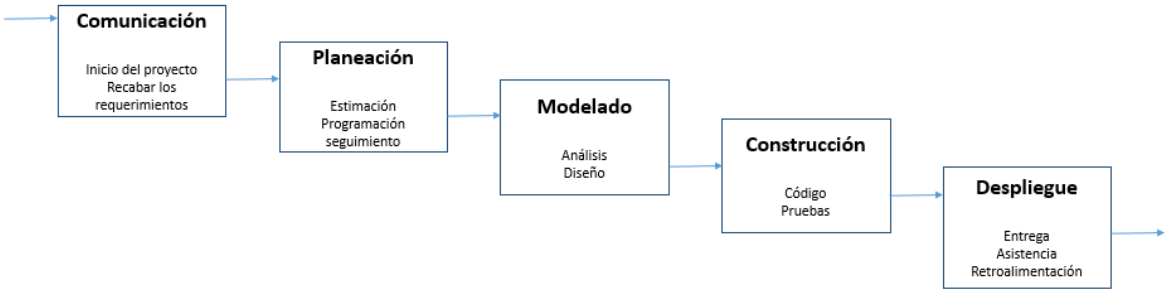


Figura 6. Modelo en Cascada (Elaboración propia).

El modelo de la cascada, a veces llamado ciclo de vida clásico, sugiere un enfoque sistemático y secuencial6 para el desarrollo del software, que comienza con la especificación de los requerimientos por parte del cliente y avanza a través de planeación, modelado, construcción y despliegue, para concluir con el apoyo del software terminado (véase la figura 6) (Pressman, 2010).

2.3.3 Modelo en V

Una variante del modelo en cascada es el Modelo en V. En la figura 7 se ilustra el modelo en V, donde se aprecia la relación entre las acciones para el aseguramiento de la calidad y aquellas asociadas con la comunicación, modelado y construcción temprana. A medida que el equipo de software avanza hacia abajo desde el lado

izquierdo de la V, los requerimientos básicos del problema mejoran hacia representaciones técnicas cada vez más detalladas del problema y de su solución. Una vez que se ha generado el código, el equipo sube por el lado derecho de la V, y en esencia ejecuta una serie de pruebas (acciones para asegurar la calidad) que validan cada uno de los modelos creados cuando el equipo fue hacia abajo por el lado izquierdo.7 En realidad, no hay diferencias fundamentales entre el ciclo de vida clásico y el modelo en V. Este último proporciona una forma de visualizar el modo de aplicación de las acciones de verificación y validación al trabajo de ingeniería inicial (Pressman, 2010).

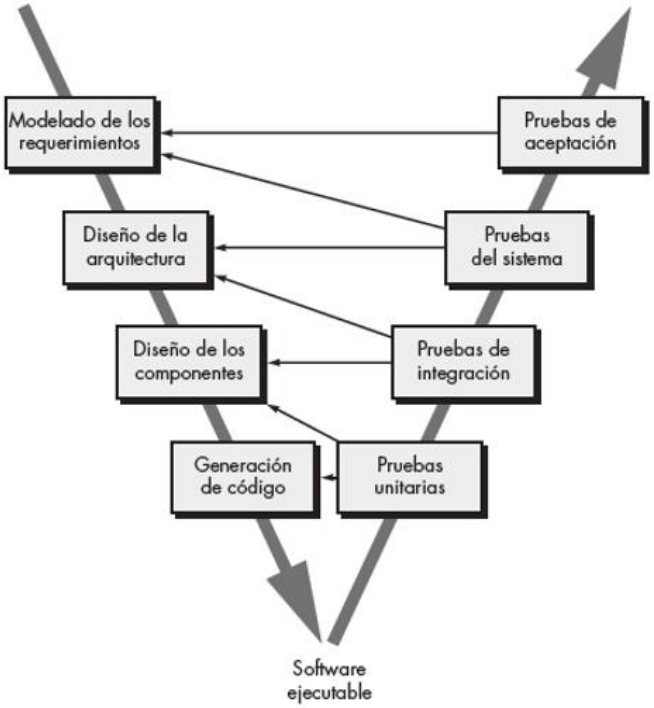


Figura 7. Modelo en V (Pressman, 2010).

2.3.4 Modelo de proceso incremental

El modelo incremental combina elementos de los flujos de proceso lineal y paralelo. En relación con la figura 7, el modelo incremental aplica secuencias lineales en forma

escalonada a medida que avanza el calendario de actividades. Cada secuencia lineal produce “incrementos” de software susceptibles de entregarse de manera parecida a los incrementos producidos en un flujo de proceso evolutivo (Pressman, 2010).

Por ejemplo, un software para procesar textos que se elabore con el paradigma incremental quizá entregue en el primer incremento las funciones básicas de administración de archivos, edición y producción del documento; en el segundo dará herramientas más sofisticadas de edición y producción de documentos; en el tercero habrá separación de palabras y revisión de la ortografía; y en el cuarto se proporcionará la capacidad para dar formato avanzado a las páginas. Debe observarse que el flujo de proceso para cualquier incremento puede incorporar el paradigma del prototipo (Pressman, 2010).

Cuando se utiliza un modelo incremental, es frecuente que el primer incremento sea el pro-ducto fundamental. Es decir, se abordan los requerimientos básicos, pero no se proporcionan muchas características suplementarias (algunas conocidas y otras no). El cliente usa el producto fundamental (o lo somete a una evaluación detallada). Como resultado del uso y/o evaluación, se desarrolla un plan para el incremento que sigue. El plan incluye la modificación del producto fundamental para cumplir mejor las necesidades del cliente, así como la entrega de características adicionales y más funcionalidad. Este proceso se repite después de entregar cada incremento, hasta terminar el producto final (Pressman, 2010).

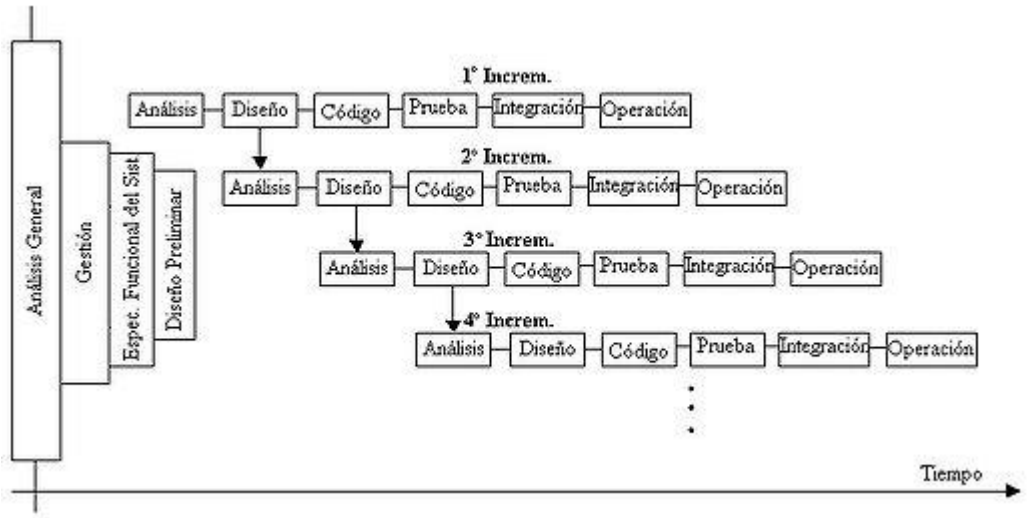


Figura 8. Modelo incremental (Pressman, 2010).

El modelo de proceso incremental se centra en que en cada incremento se entrega un pro-ducto que ya opera. Los primeros incrementos son versiones desnudas del producto final, pero proporcionan capacidad que sirve al usuario y también le dan una plataforma de evaluación (Pressman, 2010).

El desarrollo incremental es útil en particular cuando no se dispone de personal para la implementación completa del proyecto en el plazo establecido por el negocio. Los primeros incrementos se desarrollan con pocos trabajadores. Si el producto básico es bien recibido, entonces se agrega más personal (si se requiere) para que labore en el siguiente incremento. Además, los incrementos se planean para administrar riesgos técnicos. Por ejemplo, un sistema grande tal vez requiera que se disponga de hardware nuevo que se encuentre en desarrollo y cuya fecha de entrega sea incierta. En este caso, tal vez sea posible planear los primeros incrementos de forma que eviten el uso de dicho hardware, y así proporcionar una funcionalidad parcial a los usuarios finales sin un retraso importante (Pressman, 2010).

2.3.5 Modelo de proceso evolutivo

El software, como todos los sistemas complejos, evoluciona en el tiempo. Es frecuente que los requerimientos del negocio y del producto cambien conforme avanza el desarrollo, lo que hace que no sea realista trazar una trayectoria rectilínea hacia el producto final; los plazos apretados del mercado hacen que sea imposible la terminación de un software perfecto, pero debe lanzarse una versión limitada a fin de aliviar la presión de la competencia o del negocio; se comprende bien el conjunto de requerimientos o el producto básico, pero los detalles del producto o extensiones del sistema aún están por definirse. En estas situaciones y otras parecidas se necesita un modelo de proceso diseñado explícitamente para adaptarse a un producto que evoluciona con el tiempo (Pressman, 2010).

Los modelos evolutivos son iterativos. Se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software. En los párrafos que siguen se presentan dos modelos comunes de proceso evolutivo (Pressman, 2010).

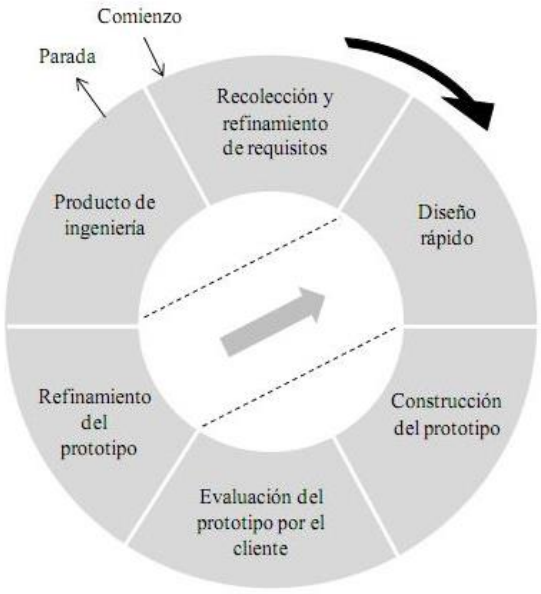


Figura 9. Modelo evolutivo (Pressman, 2010).

2.3.6 Modelo en espiral

El modelo espiral es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Tiene el potencial para hacer un desarrollo rápido de versiones cada vez más completas. Boehm describe el modelo del modo siguiente:

El modelo de desarrollo espiral es un generador de modelo de proceso impulsado por el riesgo, que se usa para guiar la ingeniería concurrente con participantes múltiples de sistemas intensivos en software. Tiene dos características distintivas principales. La primera es el enfoque cíclico para el crecimiento incremental del grado de definición de un sistema y su implementación, mientras que disminuye su grado de riesgo. La otra es un conjunto de puntos de referencia de anclaje puntual para asegurar el compromiso del participante con soluciones factibles y mutuamente satisfactorias (Boehm, 2011).

Con el empleo del modelo espiral, el software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, lo que se entrega puede ser un modelo o prototipo. En las iteraciones posteriores se producen versiones cada vez más completas del sistema cuya ingeniería se está haciendo (Pressman, 2010).

Un modelo en espiral es dividido por el equipo de software en un conjunto de actividades estructurales. Para fines ilustrativos, se utilizan las actividades estructurales generales ya analizadas. Cada una de ellas representa un segmento de la trayectoria espiral ilustrada en la figura 10. Al comenzar el proceso evolutivo, el equipo de software realiza actividades implícitas en un circuito alrededor de la espiral en el sentido horario, partiendo del centro (Pressman, 2010).

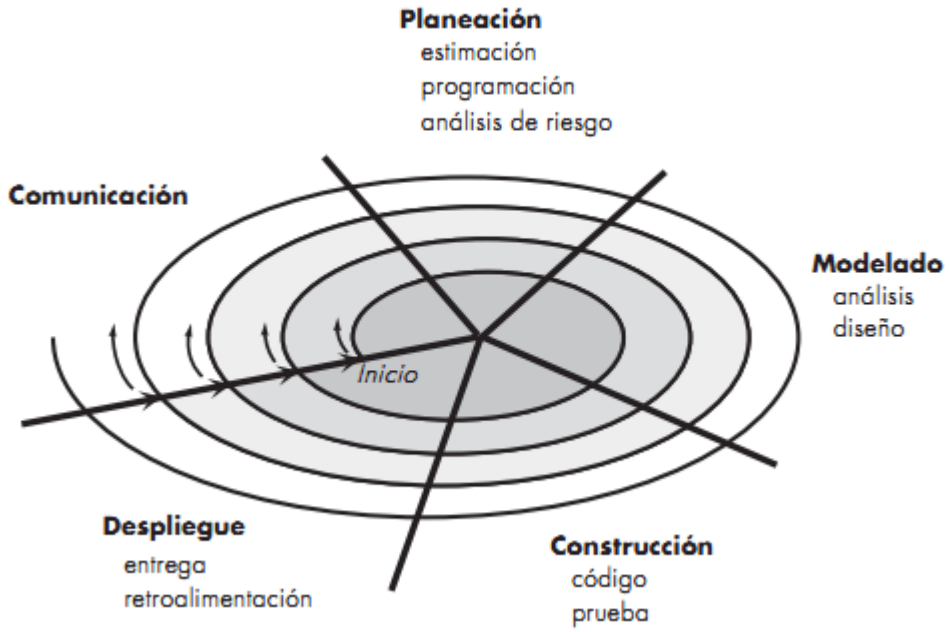


Figura 10. Modelo en espiral (Pressman, 2010).

El primer circuito alrededor de la espiral da como resultado el desarrollo de una especificación del producto; las vueltas sucesivas se usan para desarrollar un prototipo

y, luego, versiones cada vez más sofisticadas del software. Cada paso por la región de planeación da como resultado ajustes en el plan del proyecto. El costo y la programación de actividades se ajustan con base en la retroalimentación obtenida del cliente después de la entrega. Además, el gerente del proyecto ajusta el número planeado de iteraciones que se requieren para terminar el software (Pressman, 2010).

A diferencia de otros modelos del proceso que finalizan cuando se entrega el software, el modelo espiral puede adaptarse para aplicarse a lo largo de toda la vida del software de cómputo (Pressman, 2010).

El modelo espiral es un enfoque realista para el desarrollo de sistemas y de software a gran escala. Como el software evoluciona a medida que el proceso avanza, el desarrollador y cliente comprenden y reaccionan mejor ante los riesgos en cada nivel de evolución. El modelo espiral usa los prototipos como mecanismo de reducción de riesgos, pero, más importante, permite aplicar el enfoque de hacer prototipos en cualquier etapa de la evolución del producto. Mantiene el enfoque de escalón sistemático sugerido por el ciclo de vida clásico, pero lo incorpora en una estructura iterativa que refleja al mundo real en una forma más realista. El modelo espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y, si se aplica de manera apropiada, debe reducir los riesgos antes de que se vuelvan un problema (Pressman, 2010).

2.3.7 Modelo concurrente

El modelo de desarrollo concurrente, en ocasiones llamado ingeniería concurrente, permite que un equipo de software represente elementos iterativos y concurrentes de cualquiera de los modelos de proceso descritos en este capítulo. Por ejemplo, la actividad de modelado definida para el modelo espiral se logra por medio de invocar

una o más de las siguientes acciones de software: hacer prototipos, análisis y diseño (Pressman, 2010).

La figura 2.11 muestra la representación esquemática de una actividad de ingeniería de software dentro de la actividad de modelado con el uso del enfoque de modelado concurrente. La actividad “modelado” puede estar en cualquiera de los estados mencionados en un momento dado. En forma similar, es posible representar de manera análoga otras actividades, acciones o tareas (por ejemplo, comunicación o construcción). Todas las actividades de ingeniería de software existen de manera concurrente, pero se hallan en diferentes estados (Pressman, 2010).

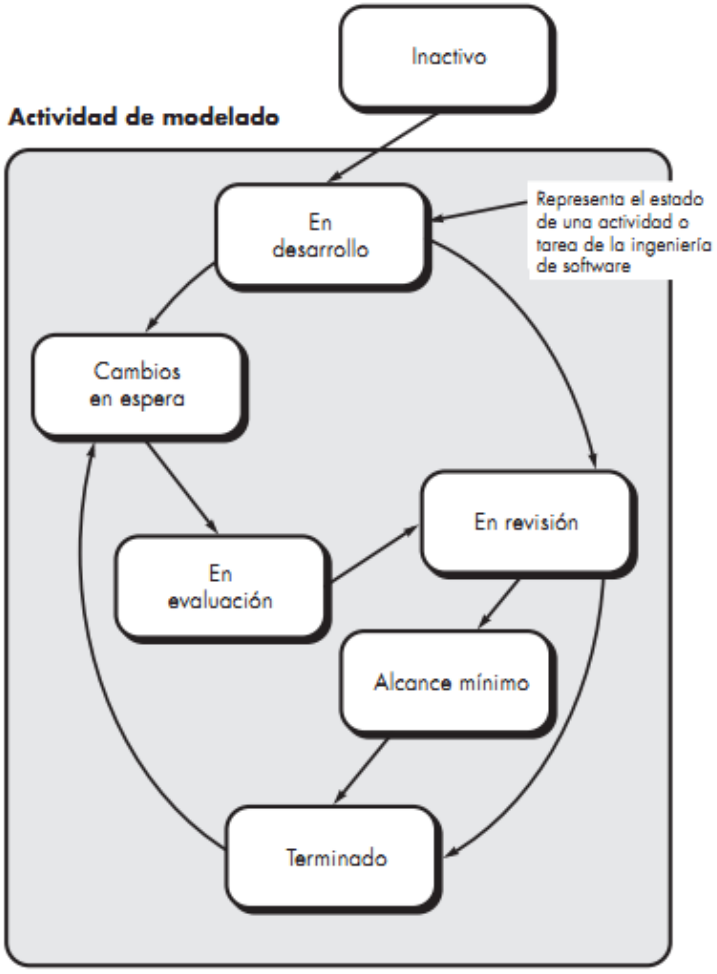


Figura 11. Modelo de proceso concurrente (Pressman, 2010).

El modelado concurrente define una serie de eventos que desencadenan transiciones de un estado a otro para cada una de las actividades, acciones o tareas de la ingeniería de software. Por ejemplo, durante las primeras etapas del diseño (acción importante de la ingeniería de software que ocurre durante la actividad de modelado), no se detecta una inconsistencia en el modelo de requerimientos. Esto genera el evento corrección del modelo de análisis, que disparará la acción de análisis de requerimientos del estado terminado al de cambios en espera (Pressman, 2010).

El modelado concurrente es aplicable a todos los tipos de desarrollo de software y proporciona un panorama apropiado del estado actual del proyecto. En lugar de confinar las actividades, acciones y tareas de la ingeniería de software a una secuencia de eventos, define una red del proceso. Cada actividad, acción o tarea de la red existe simultáneamente con otras actividades, acciones o tareas. Los eventos generados en cierto punto de la red del proceso desencadenan transiciones entre los estados (Pressman, 2010).

Capítulo 3 Desarrollo del modelo propuesto

El objetivo de este trabajo es desarrollar un nuevo modelo para la generación y ejecución de pruebas, que ayude a empresas y organizaciones desarrolladoras de software en la verificación y validación de productos de alta competitividad. Dicho supuesto podrá ser utilizado para modelos de desarrollo en espiral, cascada, evolutivos, incremental, modelo V, RUP, o los descritos en la bibliografía propuesta Sommerville, (2011).

En este capítulo se determinan y describen las fases de las que está compuesto dicho modelo y que ayudaran a cumplir con los objetivos iniciales de la investigación.

De acuerdo a las necesidades y áreas de oportunidad que se han observado en diversas metodologías propuestas por algunos autores descritos en el capítulo 1, se establecieron etapas que no solo cumplirán con un proceso definido para la ejecución de pruebas, sino también en el seguimiento de las mismas, y se presentaran plantillas que ayudaran a la generación, ejecución, seguimiento y cierre de errores para el cumplimiento de los objetivos.

Finalmente el modelo propuesto puede ser adaptado, rectificado y conducido según el fin y objetivos que cada organización y empresa establezcan, eso significa que, las herramientas o artefactos presentados en esta metodología pueden hasta cierto punto no utilizarse, o poder ser remplazada por alguna otra actividad equivalente, siempre y cuando no se omitan etapas ya que podría tener repercusiones en el proceso.

3.1 Determinación y elaboración de la metodología propuesta

El presente trabajo se concentra en la creación de un modelo para la generación y ejecución de pruebas como medio de verificación y validación de software de calidad. En base a lo mencionado anteriormente, se propone una metodología basada en tres etapas siguientes:

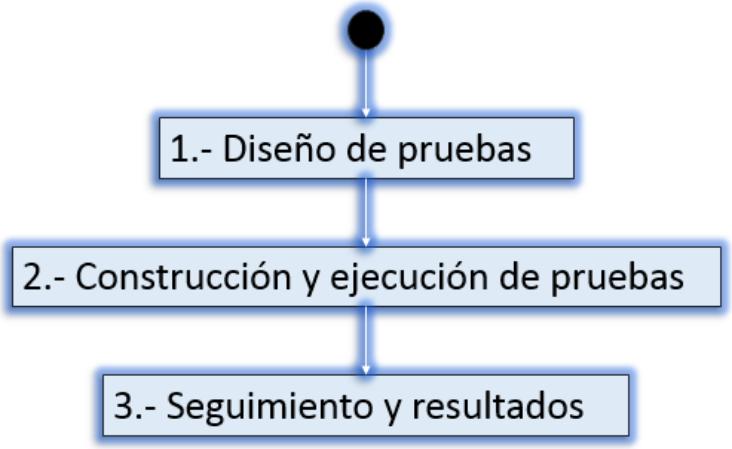


Figura 12. Etapas de metodología (Elaboración propia).

La finalidad de la creación del modelo, no solo es guiar todas las etapas de pruebas al software, también está diseñada para cumplir con los procedimientos para construir un sistema de calidad.

La eficacia es un aspecto centrado principalmente en la implementación, si la implementación sigue al diseño, y el producto resultante cumple con los objetivos de requisitos y de rendimiento, la calidad es alta.

Para poder cumplir lo dicho, existen organismos internacionales de estándares y certificación relacionados con el desarrollo de sistemas como: IEEE (Instituto de Ingeniería Eléctrica y Electrónica), CMMI (Integración de modelos de madurez de capacidades), o MoproSoft (Modelo de Procesos para la Industria del Software), descritos en el capítulo 2, en los cuales, se debe tener un proceso definido y completo

que abarque, la planificación, ejecución, seguimiento y los resultados para el cumplimiento de las actividades. Dicha propuesta cumplirá con los requisitos y objetivos que se solicita para poder obtener una certificación en alguna de las normas mencionadas, si se requiriere por parte de la organización.

Cabe mencionar que para poder adquirir una certificación por parte de estos organismos, es indispensable que todas las áreas de la empresa cumplan con los lineamientos marcados en cada uno de los estándares, por ello la importancia de cumplir con las etapas y procedimientos que presenta el modelo propuesto, con este se cumplirán dichas rubricas que al área de pruebas compete.

Es importante mencionar que los organismos de certificación mencionados, más conocidos y utilizados a nivel nacional e internacional, crean que las empresas certificadas en estos estándares, puedan tener un nivel más alto en la creación de productos de calidad, que conlleva a tener un mayor alcance y generar mayor confianza en sus clientes finales.

En la actualidad, es indispensable contar con certificaciones nacionales e internacionales, tanto individuales como organizacionales, así como contar con procedimientos bien definidos que garanticen a los clientes la producción de una aplicación de forma adecuada y correcta. Son puntos fundamentales para poder obtener un contrato en el ámbito del desarrollo de software, como sobrevivir al cambiante ámbito mercantil y tecnológico.

Por todo lo anterior, cada una de las tres principales etapas del modelo propuesto, se dividen en subetapas específicas, mediante las cuales podrá guiarse el proceso de inicio a fin, que ayudara a generar productos de calidad en comparación con todas aquellas organizaciones en las cuales no se utilizó un procedimiento en la ejecución de pruebas.

A continuación se presenta la clasificación de las tres etapas del modelo que posteriormente serán explicadas a detalle:

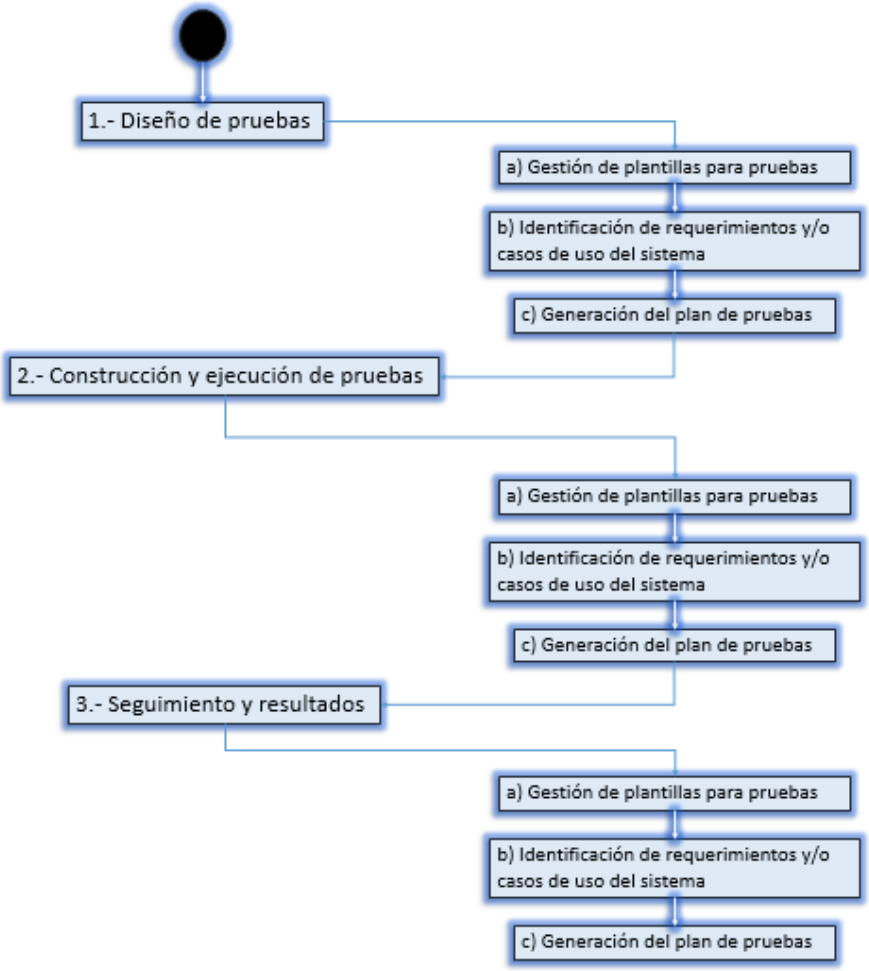


Figura 13. Etapas detalladas del modelo (Elaboración propia).

Es importante mencionar que cada una de las etapas presentadas, son un factor determinante en el cumplimiento de los objetivos descritos en esta investigación, así como los procedimientos, documentación, herramientas y apoyos brindados.

3.1.1 Diseño de pruebas

El diseño, es la primer etapa dentro de todo el proceso de ejecución de pruebas, es muy importante se lleve a cabo de forma adecuada y correcta, ya que de esto dependerá el resto de las actividades. Es indispensable que el responsable de esta actividad, tenga bien definido cuál es el objetivo general del proceso y de manera específica de este punto, para facilitar la gestión y actividades que tenga que desarrollar.

Así como es indispensable se cumpla el punto anterior, es importante seguir el proceso de forma adecuada y puntual, para encontrar el mayor número de errores dentro del sistema a probar y poder liberar un producto con las características y requisitos deseados. Si por alguna circunstancia llegase a omitir el proceso o alguna de las etapas, puede generar problemas en, tiempos de entrega, una mala comunicación, no probar toda la funcionalidad del software, delegar de forma incorrecta responsabilidades a los integrantes del equipo, así como liberar un producto de mala calidad y generar molestias al usuario o cliente final al no cumplir con las necesidades y funciones fundamentales con que fue pedida dicha aplicación.

Si bien, se especificó que el no cumplimiento de las actividades podría repercutir en no llevar a cabo un buen proceso de pruebas, cabe mencionar que dentro de las subetapas, puede haber y apoyarse de algún documento adicional que no fue contemplado en esta investigación, esto dependerá de los lineamientos y documentos que en cada organización se manejan y generan.

Para poder realizar el proceso de pruebas, es indispensable contar con los elementos y documentos base para guiar las actividades de forma adecuada, así como el procedimiento para una buena ejecución.

A continuación se describirán las tres fases de la que está compuesta la etapa de diseño de pruebas, los elementos y herramientas de las que cada actividad está formada.

3.1.1.1 Gestión de plantillas para pruebas

La gestión, es una actividad en la cual una o varias personas, se preocupan por la disposición de los recursos y estructuras necesarias para realizar una actividad, así como la coordinación de tareas para el cumplimiento de los objetivos.

En toda actividad referente al desarrollo de software, debe existir documentación que respalde no solo los requisitos básicos de la actividad, si no también, los archivos que en cada etapa se van generando. La importancia de la acción radica, en más que una buena práctica por parte de cada área, también es permitir que la información registrada en colecciones de datos sea accesible eficientemente, además que sea almacenada de forma segura y conservada por el periodo de tiempo necesario, para una futura consulta y permita ayudar a la toma de decisiones.

Un punto adicional y muy importante de la documentación, es proveer al usuario final de escritos que avalen las actividades realizadas, en el caso muy particular de pruebas, ayuda al usuario final a validar el procedimiento ejecutado para probar el software antes de ser liberado y las herramientas utilizadas, logrando con esto, no solo la confianza del consumidor final, si no también, respaldar la empresa por sus procedimientos que lleva a cabo, y las buenas practicas que son parte fundamental de las actividades en la organización.

Por ello, el primer paso de todo el proceso de ejecución de pruebas, es tener las herramientas básicas para poder iniciar con la documentación y planeación necesaria. En este aparto, el probador pedirá al líder del área de pruebas las plantillas necesarias que se ocuparan en el proceso, mismas que serán entregadas al cliente final para su verificación y validación.

A continuación se mencionan los documentos y artefactos base que ocupa la metodología propuesta:

- Plan de pruebas
- Matriz de datos para pruebas

- Cronograma de actividades
- Diagrama del ciclo de vida de errores
- Bugzilla

Con los elementos que se mencionan, se podrá realizar de forma satisfactoria todo el proceso de pruebas de inicio a fin, ya que cuenta con las herramientas necesarias y procesos descritos para una correcta aplicación. Dichos instrumentos se encuentran para su consulta a detalle en el apéndice de esta investigación (Ver apéndice A, B, C, D), sin embargo, en etapas posteriores se describirán a detalle.

Es importante mencionar que la documentación cuenta con información general que ayudara a la creación, ejecución, seguimiento y cierre de pruebas, logrando reducir el tiempo que se destina en esta actividad. Así mismo, cabe mencionar que dicha información puede ser manipulada a conveniencia de la empresa, organización o grupo, según sus fines para cumplir sus objetivos establecidos.

A continuación se describe de forma general la función y definición de cada documento para su comprensión, posteriormente será descrito a detalle en base a la metodología propuesta.

Plan de Pruebas.

En todo proceso de cualquier área, es indispensable tener un archivo que guie las actividades que se llevan a cabo, este caso no es la excepción. Este documento regirá a todo el proceso de pruebas de inicio a fin, se describe la forma en cómo se llevara a cabo el proceso de pruebas para garantizar el éxito. Un plan debe incluir factores que ayuden a guiar todo el proceso e involucrados, esto desde el momento que se consideró la construcción de sistemas de información una necesidad para suplir procesos ha sido notoria la exigencia de productos con calidad.

Un plan de pruebas, es un conjunto de actividades que se enmarcan en la identificación de fallas de un sistema de información. Ofrece al equipo de pruebas y desarrollo, información sobre las tareas que deberán ejecutarse en cada módulo, las responsabilidades y los recursos designados para tal fin.

Este documento incluye puntos y aspectos como:

- Un identificador.
- Un alcance.
- Elementos a probar.
- Estrategia a utilizar.
- Categorización.
- Archivos a entregar al final del proceso.
- Recursos a utilizar para dicha actividad.
- Calendario de tareas.
- Control o manejo de riesgos.
- Responsables o involucrados de las áreas.

En base a lo escrito, el documento propuesto y diseñado está compuesto por cuatro secciones. La primera sección de información general, con fechas y revisiones que se tendrán que realizar por cada iteración al archivo. La segunda parte es información básica, se describe y especifica información que será de gran ayuda para todo el proceso. Un tercer apartado llamado información técnica, con datos referentes y específicos de artefactos, criterios de ejecución y la forma como se procederá a lo largo de las pruebas. Finalmente una cuarta, con firmas correspondientes al equipo de trabajo e involucrados de las dos partes, que indicaran y firmaran cuando el proceso haya llegado a su fin con resultados satisfactorios para ambas partes.

Matriz de datos para pruebas

En todo proceso de pruebas, debe de existir un documento en el cual se referencien todos los experimentos que se realizaron al software, así mismo cada prueba debe tener un identificador, la finalidad de este último es poder llevar un seguimiento del mismo de inicio a fin, así como saber de qué artefactos o documentos depende y la relación con estos. Existe una gran diversidad en la forma de cómo crear o realizar esta actividad, ya que podría ser mediante documentos físicos, digitales, o herramientas especializadas para la creación de pruebas.

En el caso particular de la metodología, incluye, pruebas funcionales, pruebas integrales, pruebas de seguridad, pruebas de rendimiento y pruebas de carga, las cuales estarán documentadas mediante un archivo donde se registran todos los elementos a probar, el procedimiento y ruta para ejecutar cada uno de los experimentos. Es importante tener dicho archivo, ya que cuando el líder de pruebas valida dicho escrito, verifica que cada uno de los componentes de la aplicación fueron probados, y no solo eso, si no también identificar con que información se realizaron los procedimientos. Esta táctica tiene dos fines, identifica todos los errores existentes en la aplicación y respaldar el trabajo realizado en el área de pruebas. Si en la revisión, alguna táctica en la ejecución de pruebas fue realizada de forma incorrecta, esta se tendrá que repetir, si alguna prueba fue omitida tendrá que diseñarse y aplicarse, con la finalidad de no omitir alguna función a probar del software.

Así, el objetivo primordial de este archivo, es proveer de información al usuario final, al área de pruebas y área de desarrollo, así como a la administración, respecto a las pruebas que le fueron realizadas al sistema antes de haberse liberado. Esta parte es un punto fundamental que debe llevarse a cabo en todas las empresas de desarrollo, y formar una buena práctica a los integrantes del área.

Cronograma de actividades

También conocido como calendario de trabajo, es una de las actividades muy importante en la gestión de proyectos. Su principal función es identificar y definir cada una de las tareas que lo componen y distribuir temporalmente su realización. Si bien se mencionó que el plan de pruebas será el artefacto que guiara todo el proceso de pruebas, este requiere de un cronograma que pueda gobernar todas las actividades con fechas y tiempos específicos.

Dicho documento ya sea impreso o digital, debe incluir una lista de actividades o tareas con fechas previstas de su comienzo a fin. Para poder realizar esta tarea, existe una gran variedad de programas informáticos, los cuales pueden servir de utilidad, entre los que están, Word, Excel o algunos más especializados como, Open Project con licencia gratuita, y Microsoft Project en la cual se requiere de una licencia de paga para poder utilizarla.

Para el caso particular de esta actividad, se tomaron en cuenta algunos factores relevantes a la hora de su creación, estos son: revisiones, validaciones y cierre, utilizando la herramienta Microsoft Project.

Ciclo de vida de errores

Uno de los procesos que debe estar bien definido, es el seguimiento y proceso de solución de errores. Podemos considerar un desacierto al fallo en un programa de computador o sistema de software que desencadena un resultado indeseado. Así mismo, podríamos decir, que un ciclo es una actividad repetitiva durante un determinado tiempo, y que en algún momento vuelve a la configuración inicial.

Entonces, podemos concluir que el ciclo de los errores, será el proceso que llevara a un defecto desde el momento en que es registrado, hasta el momento que es solucionado. Este punto es uno de los temas determinantes en la creación de un

sistema, debido a que si no se tiene claro cuál es el curso que tiene que llevar esta acción, puede repercutir en los tiempos de entrega.

Es muy importante mencionar que cada empresa debe tener identificados los actores y definido el proceso que debe llevar a cabo en el ciclo de vida de un defecto, esta actividad en la actualidad no tiene mucha relevancia en las organizaciones, por ello que el software creado, contiene una gran cantidad de defectos cuando sale al mercado.

Estudios han arrojado que, aquellas empresas las cuales destinan tiempo en un buen proceso de pruebas, logran generar productos con un mínimo de errores e incluso no tener la necesidad de realizar un mantenimiento posterior. Por ello la importancia que exista un área especializada para llevar a cabo esta actividad.

Por todo lo anterior, la metodología propuesta, presenta actividades fundamentales que deben cumplirse y plasmarse en el cronograma de actividades, así mismo se debe dar el cumplimiento de estas al pie de la letra, y con el flujo predeterminado, para garantizar que los errores encontrados sean corregidos en tiempo y forma.

Bugzilla

En la actualidad, existe una gran gama de programas para el seguimiento de errores, su principal función es, proveer de información necesaria para mantener la calidad del software, sobre todo durante la etapa de desarrollo. Entre las aplicaciones para este fin esta, Mantis, Trac, GNATS, Roundup, Scarab, por mencionar algunas. La mayoría de estas son de software libre y su utilización dependerá de la empresa en base a sus objetivos y políticas.

Para el caso particular de este modelo, se utilizara la herramienta *Bugzilla*, ya que es un instrumento basado en Web de seguimiento de errores (*Bug Tracking System* o BTS, por sus siglas en inglés), originalmente desarrollada y usada por el proyecto

Mozilla, lanzado como software de código abierto por *Netscape Communications* en 1998, *Bugzilla* ha sido adoptado por una variedad de organizaciones para su empleo en el seguimiento de defectos (errores), su licenciamiento es bajo la licencia pública de Mozilla (Bugzilla, 2015).

Como nos podemos dar cuenta, las bases con las cuales se desarrolla la metodología propuesta, están muy bien determinadas, el grado de no solo indicar la característica de cada una de ellas, sino también, presentar el objetivo y razón principal por la cual se utilizara una determinada herramienta. En las secciones posteriores, se detalla a fondo, cada uno de los componentes mencionados en este punto.

3.1.1.2 Identificación de requerimientos y casos de uso del sistema

Una vez que se han gestionado los documentos base que se ocuparan en las fases siguientes, el subsecuente paso es identificar los artefactos que se probaran para validar la complejidad y estimar el tiempo de entrega.

Para ello se propone utilizar los documentos que se generaron en las primeras fases de análisis y diseño del sistema, que dependerán de cada empresa u organización, obedeciendo a sus lineamientos serán los entregables que se tengan, así como los nombres que se le den. Sin embargo a continuación se describe algunos de los documentos que servirán para identificar los módulos que se probaran.

- Documento que contenga el análisis de requerimientos aprobado por el cliente, en el cual debe estar especificado todos los módulos, componentes y funciones que debe cumplir la aplicación, así como el alcance para evitar futuros mal entendidos.
- Escritos que contengan los casos de uso con la descripción de cada elemento que contendrá el sistema. Si bien los requisitos podrían ser un poco generales

en algunos casos, los casos de uso tiene como finalidad brindar al desarrollador como debe funcionar de inicio a fin una actividad, involucrando actores para una mejor comprensión.

- Diagramas de navegación, secuencias, y/o de clases que ayudaran a entender lo que se requiere entregar al cliente, son de gran ayuda gracias a la especificación del flujo en cada una de las funciones y operaciones a realizar, la forma en como están relacionadas las partes con el resto y su agrupación en general
- Archivo de diseño del sistema para comprender como estará compuesto el proyecto, proporcionará un panorama previo al funcionamiento final de la aplicación, si bien no es funcional en su totalidad, el menos brindara la forma en cómo se llevara a cabo la navegación.

Con los documentos anteriores será fácil la identificación de los elementos que se probaran en la fase de pruebas, sin embargo, no siempre se tienen todos los archivos para su consulta, por ello es importante que el probador o la persona encargada de realizar esta tarea, tenga bien entendido lo que el sistema tiene que realizar. Se recomienda que dicha persona esté en las juntas que se llevan a cabo por parte del equipo que levanta los requisitos la aplicación y el cliente, así le será fácil identificar los componentes a probar desde etapas tempranas.

3.1.1.3 Generación del plan de pruebas

En todo proceso dentro de la ingeniería de software se debe tener un plan en cual guiara todas las actividades para llegar a un fin. En el área de pruebas no es la excepción, por lo cual se propone un plan el cual tiene como uno de sus objetivos, describir las herramientas y técnicas con el máximo estándar de calidad y confiabilidad,

para unificar los procesos que garanticen las mejores prácticas en la liberación de software.

Dicho plan como se menciona, tiene el objetivo primordial de gobernar todas las etapas de pruebas, sin embargo, es importante describir sus objetivos puntuales que ayudaran en el diseño y ejecución de pruebas que sistemáticamente descubran los errores con el menor tiempo, recursos y esfuerzo posible para el cumplimiento de las tareas.

Los principales objetivos de un plan de prueba son:

- Definir los tiempos que se destinaran a las pruebas.
- Describir los componentes que van a ser probados.
- Indicar los materiales, herramientas y recursos humanos que se utilizaran para la ejecución de los experimentos.
- Brindar un reporte final respecto a los ensayos realizados y el porcentaje de satisfacción.
- Ser guía de todo el proceso de pruebas de inicio a fin.

Todo lo anterior, para establecer un marco de trabajo y verificar que los productos funcionen de acuerdo a lo esperado y especificado que se describió en la etapa de requerimientos, así como la forma y medios para registrar los resultados de la ejecución de pruebas. Dentro de la metodología propuesta también es importante describir el alcance que tendrá y el tipo de prueba que abarcara dicho modelo.

Con la finalidad de probar el sistema, es necesario realizar una seria de pruebas distintas que permitan asegurar que los requerimientos se cumplan, así como, la forma en la que se ejecutarán, los criterios de aceptación, la forma y medios para registrar los resultados de la ejecución de los experimentos. En seguida se listan los cinco tipos de pruebas que abarca el modelo.

- Pruebas funcionales.
- Pruebas integrales.
- Pruebas de seguridad.
- Pruebas de carga.
- Pruebas de rendimiento.

En el plan de pruebas también se definen aspectos como, documentos que se entregaran al final del proceso, las herramientas que se necesitan, las versiones, los archivos de apoyo que ayudaran al objetivo del plan, el cronograma de trabajo que seguirá el equipo para cumplir en tiempo y forma las tareas determinadas, los riesgos que podrán surgir en algún momento de esta etapa, la clasificación y severidad de los errores, los criterios para ejecución, suspensión, reanudación, aceptación y validación en los experimentos. Finalmente describe el plan de comunicación, el escalamiento de incidencias, algunas definiciones, acrónimos y abreviaturas que ocupa el documento (Ver apéndice A).

Finalmente todo documento debe tener un nombre para su identificación y seguimiento, a continuación se realiza una propuesta para el plan de pruebas.

[Acrónimo del proyecto] - Plan de Pruebas

En donde:

Acrónico del proyecto: Hace referencia a las siglas que lo identifican, ya sea la primera letra del nombre completo de la aplicación o bien algún calificativo según su función, esto dependerá de los primeros documentos creados donde vendrá descrito ese aspecto, caso contrario el área de pruebas tendrá que gestionar dicho acrónimo para su uso.

En seguida se describe a detalle los apartados de los que está compuesta cada sección del documento plan de pruebas para su completa comprensión y correcto uso.

SECCIÓN I: CONTROL

Revisiones

Este apartado está hecho para validar las revisiones que se tendrán a lo largo del proceso de pruebas, mediante la tabla 1, en dicho documento se especificaran responsables para revisión y aprobación del plan, así como las fechas en las que se llevan a cabo.

Realizo		**Fecha**	
Aprobó		Fecha	

Tabla 1. Información de revisiones (Elaboración propia).

Versiones

Lugar donde se especifican las veces en las que el documento ha sido modificado por alguna persona del equipo, así mismo, se debe definir la fecha de cada alteración. En la versión del documento actual se debe tomar en cuenta para modificaciones simples, la numeración será de .1 en lo sucesivo, mientras que para alteraciones relevantes e importantes, será de 1 en lo gradual, finalmente listar los involucrados de este documento y una breve descripción de la modificación que se realizó, esto para futuras consultas, ver tabla 2 como ejemplo de este apartado.

Versión	Descripción	Autor	Fecha

Tabla 2. Información de versiones (Elaboración propia).

SECCIÓN II: INFORMACIÓN ESENCIAL

Introducción

Como su nombre lo dice, describe un panorama general e importancia de las pruebas de software, así como su propósito que este tiene para el cumplimiento del proceso de forma satisfactoria.

A continuación se describe la introducción utilizada para el plan de pruebas:

"Una de las principales inquietudes en el área de Pruebas es la forma en cómo se aplicaran a los componentes de cada software, así mismo las herramientas necesarias para llevar acabo dichas tareas. El contenido de este documento es parte integral de una metodología de pruebas para la creación y ejecución como medio de validación de productos de calidad.

El propósito de este documento el cual contiene el Plan de Pruebas, es regir y planificar todas las etapas de las pruebas funcionales, integrales, seguridad, rendimiento y carga. El plan no solo gobierna todas las etapas, si no también establece las técnicas, herramientas y actividades relacionadas con la ejecución y validación de cada una de las etapas, incluyendo responsabilidades de cada una de las actividades, los recursos y los prerrequisitos que deben ser considerados en el esfuerzo de cada una de ellas; esto con la finalidad de llevar un correcto seguimiento de inicio a fin de este proceso y garantizar la operatividad y funcionalidad de la solución desarrollada en base a los requerimientos planteados".

Objetivos

Un objetivo es el fin al que se desea llegar, por ello se describen tanto el objetivo general como particular de las pruebas y el plan de pruebas, mismos que podrán ser modificados según el propósito que tenga cada empresa u organización. Sin embargo, se presentan algunos ejemplos que servirán de guía para este fin.

En seguida se describe el objetivo general y los específicos descritos en el plan de pruebas:

"*Objetivo general*

El presente documento tiene como objetivo diseñar, ejecutar, dar seguimiento y guiar las pruebas del software para descubrir la mayor cantidad de errores en el menor tiempo y esfuerzo posible.

Objetivos específicos:

Los principales objetivos de realizar las pruebas son:

- *Probar la funcionalidad por módulos del software conforme los escenarios definidos.*
- *Probar la funcionalidad integral del software conforme los escenarios definidos.*
- *Probar la funcionalidad de la seguridad del software conforme los escenarios definidos.*
- *Probar si el software hace lo que debe, o si provoca efectos indeseados.*
- *Encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles.*
- *Mostrar hasta qué punto las funciones del software operan de acuerdo con las especificaciones y requisitos del cliente.*
- *Establecer el alcance de las pruebas que se realizarán según su tipo y tiempo.*
- *Determinar los grupos que van a participar en la creación y ejecución de las Pruebas.*
- *Indicar los criterios de aceptación del software en la ejecución de las Pruebas.*
- *Definir herramientas que se utilizarán en la ejecución de los diferentes tipos de pruebas.*
- *Especificar la forma y medios para registrar los resultados de la ejecución de pruebas".*

Alcance

En todo documento, proceso o actividad, se debe definir hasta qué punto cubrirá la actividad. En el caso de pruebas, la finalidad es especificar los tipos de experimentos

que serán ejecutados por el área. Así mismo se describen las etapas y subetapas del modelo propuesto para el cumplimiento de los objetivos. En el siguiente texto se presenta el texto perteneciente al alcance del plan de pruebas.

"El presente documento lista las distintas pruebas a ser ejecutadas durante el proceso de prueba del sistema que permitirán asegurar el cumplimiento de los objetivos y los requerimientos del software, la forma en la que se ejecutaran, los criterios de aceptación, así como la forma y medios para registrar los resultados de la ejecución de pruebas.

Dentro de los alcances de las pruebas al sistema se establecen los siguientes puntos:

- *Pruebas Funcionales.*
- *Pruebas Integrales.*
- *Pruebas de seguridad.*
- *Pruebas de carga.*
- *Pruebas de rendimiento.*

Así mismo cabe mencionar en base a lo anterior, dicho plan forma parte integral de la metodología basada en tres etapas:

1. *Diseño de pruebas.*
2. *Construcción y ejecución de pruebas.*
3. *Seguimiento y resultados.*

Cada una de las tres principales etapas mencionadas se divide en subetapas que apoyaran al cumplimiento de los objetivos de la metodología propuesta. A continuación se describe la clasificación de cada una de ellas:

1. *Diseño de pruebas.*
    1. *Gestión de plantillas para pruebas.*

2. *Identificación de requerimientos y casos de usos del sistema.*
3. *Generación de Plan de pruebas.*

2. *Construcción y ejecución de pruebas.*
    1. *Identificación, clasificación y eliminación de casos de prueba redundantes.*
    2. *Creación de casos y escenarios de prueba al sistema.*
    3. *Ejecución de casos y escenarios de prueba al sistema.*
3. *Seguimiento y resultados.*
    1. *Reporte y seguimiento de incidencias encontradas en la ejecución.*
    2. *Cierre y resultados finales de incidencias encontradas.*
    3. *Generación de reporte final y análisis.*

Finalmente es importante señalar que cada una de dichas pruebas serán aplicadas a los módulos correspondientes o al sistema en general, esto teniendo un enfoque de pruebas de caja negra, donde se analizaran la entrada de los datos y las salida de la información para comparan con los objetivos preestablecidos".

Fuera del alcance

Apartado donde se especifican todos aquellos artefactos, componentes o experimentos que no se llevaran a cabo por parte del área de pruebas.

Por ejemplo, en seguida se listan las pruebas que no contempla el modelo propuesto:

- Pruebas de stress.
- Pruebas de caja blanca.
- Prueba de desempeño.
- Pruebas de estilo.
- Pruebas de completitud.
- Pruebas de interfaz.

Es importante mencionar, si la metodología propuesta no abarca las pruebas arriba listadas, no indica que no pueda utilizarse para ejecutarlas, podría hasta cierto punto adaptarse el documento matriz de datos para registrar los escenarios, así como el plan de pruebas para regir todo el proceso.

Modelo de pruebas

En este apartado se detalla la secuencia que llevara a cabo el proceso de experimentación, incluyendo la planificación, el ambiente y la documentación de los hallazgos encontrados.

En seguida se describe el modelo de pruebas descrito en el plan:

"La estrategia de pruebas está basada en el proceso especificado en la investigación "Modelo para la generación y ejecución de pruebas como medio de verificación y validación de productos de software de calidad" para todas aquellas empresas enfocadas al desarrollo de software, donde se definen los lineamientos y procedimientos, así como el análisis y justificación de dicho método para asegurar el éxito de los componentes y productos a validar, cuyas actividades macro son:

- *Planificación de las pruebas.*
- *Seleccionar los componentes y productos a probar.*
- *Realizar los escenarios de prueba con su correspondiente información a implementar.*
- *Establecer el ambiente para las pruebas.*
- *Establecer la versión del software a probar.*
- *Ejecutar las pruebas.*
- *Documentar los defectos, hallazgos y no conformidades detectadas en la aplicación o sistema.*

- *Analizar la resolución de defectos, hallazgos y no conformidades.*
- *Realizar el seguimiento y cierre de los errores encontrados en la ejecución.”*

SECCIÓN III: ADMINISTRACIÓN

Documentos base

Se mencionaran todos aquellos archivos o escritos que sirvieron de soporte para la creación del plan de pruebas, así como para la matriz de escenarios. Ejemplo de algunos archivos es, documento de análisis de requisitos, diagramas de casos de uso, diseño del sistema, historia del cliente y minutas, cabe recalcar que estos dependerán de la empresa y que tendrán que registrase en la tabla 3.

Nombre del documento	Versión

Tabla 3. Documentos base (Elaboración propia).

Escritos a entregar

En esta sección se especifican todos los documentos que serán entregados por el área de pruebas al finalizar el proceso, tales como, el plan de pruebas, la matriz de datos, el cronograma de actividades, el ciclo de vida de los errores, entre otros. Dichos archivos serán tantos como el área crea pertinente, es decir podrá agregar si así lo desea, para ello se utilizara la tabla 4.

Nombre del documento	Fecha de entrega	Entregado

Tabla 4. Escritos a estregar (Elaboración propia).

Participantes

Dentro de toda la actividad, existen participantes por parte del área de pruebas y de desarrollo, o incluso por parte del cliente, es importante especificar quien será el líder, así como los colaboradores de las mismas. La importancia radica en determinar responsabilidades, así como la participación de cada uno de los involucrados para la entrega de documentación y/o artefactos. En la tabla 6, se muestra la forma en como presentar dicha información.

Rol	Nombre	Participación/Actividad

Tabla 5. Participantes del área de pruebas (Elaboración propia).

Recursos materiales

En este apartado se describirán y especificaran todos los artefactos y herramientas que se utilizaran en el proceso de pruebas, es importante indicar la cantidad, el recurso y la disponibilidad del mismo, con la finalidad de hacer llegar dicha información al líder del área y este a su vez al responsable del proyecto para una correcta gestión del material, ver tabla 6.

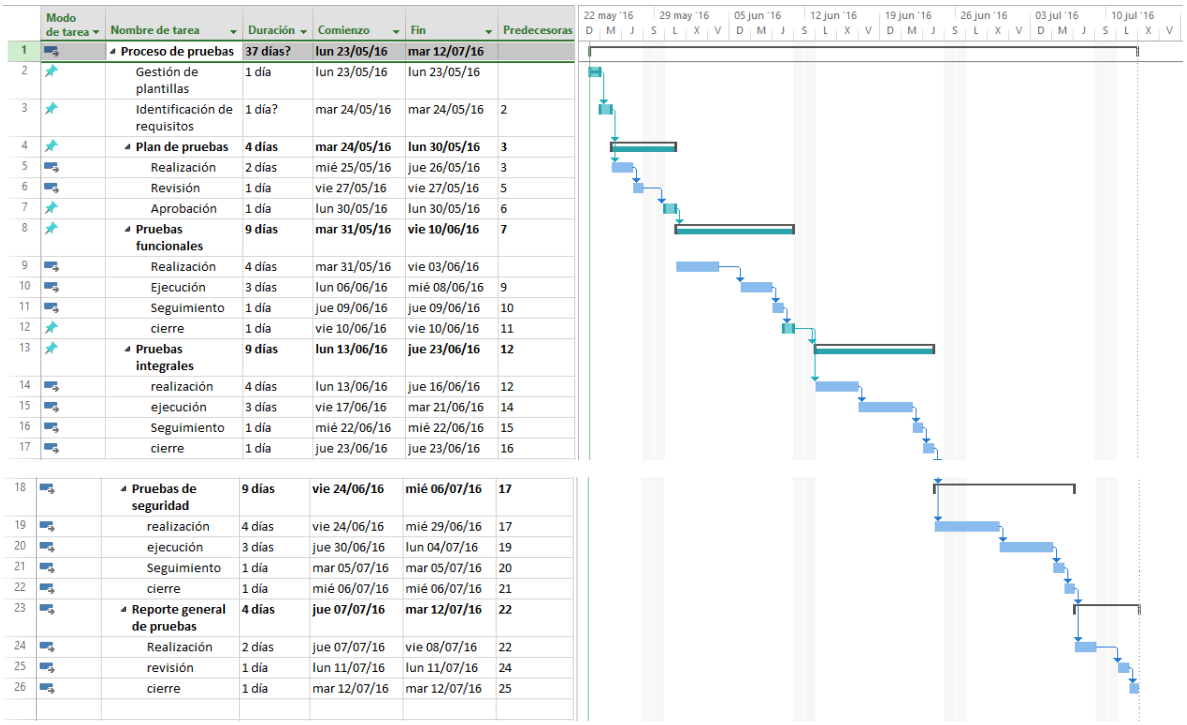
Cantidad	Recursos	Disponible

Tabla 6. Recursos materiales (Elaboración propia).

Cronograma de actividades

Se describirá toda la etapa de pruebas de inicio a fin, incluyendo cada una de las revisiones, aprobaciones y cierre para cada fase, se recomienda tener tiempos muy específicos para cada actividad, así como el cumplimiento de los mismos para evitar retrasos.

A continuación se presenta un ejemplo de lo esperado en un cronograma, así como las etapas que este debe llevar para un correcto encause del modelo de pruebas propuesto. Cabe mencionar que para este paso solo se tomaron en cuenta cinco tipos de prueba, funcionales, integrales, seguridad, carga y rendimiento.



	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**Proceso de pruebas**	**37 días?**	**lun 23/05/16**	**mar 12/07/16**	
2		Gestión de plantillas	1 día	lun 23/05/16	lun 23/05/16	
3		Identificación de requisitos	1 día?	mar 24/05/16	mar 24/05/16	2
4		**Plan de pruebas**	**4 días**	**mar 24/05/16**	**lun 30/05/16**	**3**
5		Realización	2 días	mié 25/05/16	jue 26/05/16	3
6		Revisión	1 día	vie 27/05/16	vie 27/05/16	5
7		Aprobación	1 día	lun 30/05/16	lun 30/05/16	6
8		**Pruebas funcionales**	**9 días**	**mar 31/05/16**	**vie 10/06/16**	**7**
9		Realización	4 días	mar 31/05/16	vie 03/06/16	
10		Ejecución	3 días	lun 06/06/16	mié 08/06/16	9
11		Seguimiento	1 día	jue 09/06/16	jue 09/06/16	10
12		cierre	1 día	vie 10/06/16	vie 10/06/16	11
13		**Pruebas integrales**	**9 días**	**lun 13/06/16**	**jue 23/06/16**	**12**
14		realización	4 días	lun 13/06/16	jue 16/06/16	12
15		ejecución	3 días	vie 17/06/16	mar 21/06/16	14
16		Seguimiento	1 día	mié 22/06/16	mié 22/06/16	15
17		cierre	1 día	jue 23/06/16	jue 23/06/16	16
18		**Pruebas de seguridad**	**9 días**	**vie 24/06/16**	**mié 06/07/16**	**17**
19		realización	4 días	vie 24/06/16	mié 29/06/16	17
20		ejecución	3 días	jue 30/06/16	lun 04/07/16	19
21		Seguimiento	1 día	mar 05/07/16	mar 05/07/16	20
22		cierre	1 día	mié 06/07/16	mié 06/07/16	21
23		**Reporte general de pruebas**	**4 días**	**jue 07/07/16**	**mar 12/07/16**	**22**
24		Realización	2 días	jue 07/07/16	vie 08/07/16	22
25		revisión	1 día	lun 11/07/16	lun 11/07/16	24
26		cierre	1 día	mar 12/07/16	mar 12/07/16	25

Figura 14. Cronograma de actividades (Elaboración propia).

SECCIÓN IV: DATOS TÉCNICOS

Elementos a probar

En este apartado deben mencionarse todos aquellas unidades o módulos de los que está compuesto el sistema y deben probarse, con ello se verifica y valida que el software cumpla con las características necesarias para un correcto funcionamiento, en este apartado se debe especificar el componente que se va a probar, el tipo de experimento que se le va a ejecutar, el resultado parcial y final después del seguimiento en donde se indicaran con números y porcentajes dichas consecuencias, y el cierre del proceso de pruebas determinando si el módulo fue o no aprobado. Es importante mencionar que las iteraciones en la ejecución de pruebas, serán tantas como sean necesarias, esto hasta finalizar el cierre de cada uno de los errores encontrados, a continuación se presenta la tabla 7 para este fin.

Primera Iteración:

Componentes a probar	Tipo de prueba a ejecutar	Bugs / Sugerencias / Total de escenarios	Aprobado

Tabla 7. Elementos a probar (Elaboración propia).

Control de riesgos

Es importante en toda planeación, tomar en cuenta los riesgos que pudiera surgir a lo largo del proceso, por ello en este apartado se deben especificar y describir todos aquellos conflictos que pudieran presentarse, con ello se asegura tener una acción de mitigación para poder solventar los problemas que pudieran surgir. Los datos que se deben llenar serán, un identificador del riesgo para poder llevar su control y seguimiento, una fecha correspondiente al día que se registró el suceso, la severidad demostrada en alta, media o baja, determinada por el criterio del probador, una descripción de la solución al problema, y finalmente si llegara a darse el caso, la fecha de cuando fue corregido. Ver tabla 8 para su registro.

Id	Fecha de descubrimiento	Severidad	Solución	Fecha de solución

Tabla 8. Elementos a probar (Elaboración propia).

Niveles de defectos

Aquí se describe detalladamente la clasificación de los defectos y el impacto que tienen sobre el proyecto, con ello se asegura que cada uno de estos, se dé su importancia y seguimiento correspondiente hasta la corrección del mismo.

A continuación se describe la severidad de los defectos para su clasificación, esto con la finalidad de determinar el grado al que pertenece y su correspondiente acción de corrección.

Dentro de los defectos se encuentras los siguientes:

Critica:

- Impide visualizar los resultados esperados de la prueba por alguna falla en el sistema o en la configuración.
- Existen enlaces rotos, inválidos o ausentes.
- El sistema no responde y no hay alternativas para ejecutar las pruebas.
- El sistema se cierra inesperadamente.

Alta:

- No cumple con los requerimientos.
- Inconsistencia de datos.
- Secciones de un requerimiento faltantes.
- Mal funcionamiento de algún método o complemento.

Media:

- Campos o pantallas faltantes.
- Errores que permitan continuar con el proceso.
- No muestra información completa y correctamente.
- El sistema no envía mensajes al usuario.

Baja:

- Errores ortográficos o de diseño.
- El sistema no indica que son campos obligatorios con (*) o notación resaltada.

Ciclo de vida de los defectos

En esta sección de manera gráfica se describe todo el proceso que sigue un error, desde el momento en que es descubierto hasta su cierre por solución o por ser considerado no defecto. Así mismo, se detallan las áreas participantes y el flujo que debe seguir la información, ver figura 15.

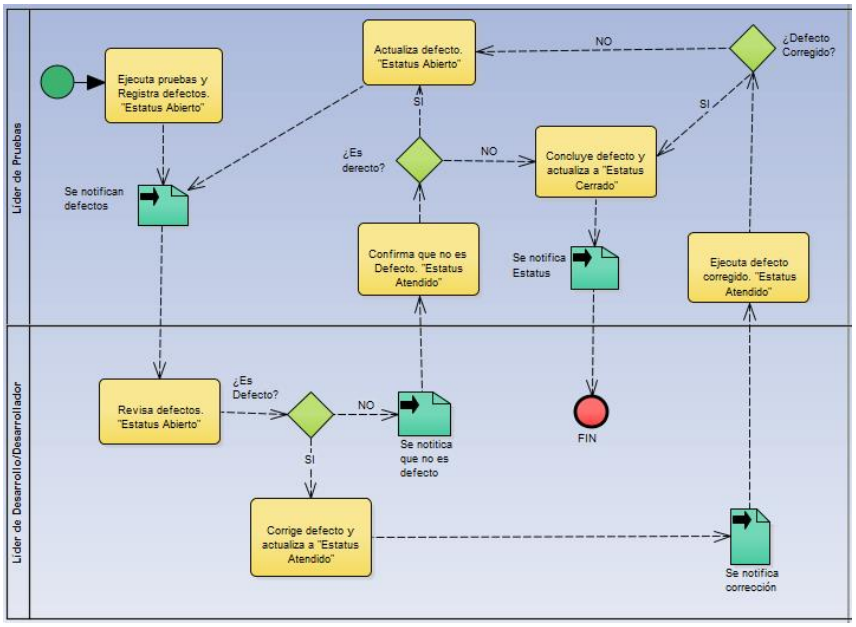


Figura 15. Ciclo de vida de errores (Elaboración propia).

Ejecución de pruebas

Una parte fundamental para la ejecución de las pruebas es este apartado, donde se describen las características mínimas que debe haber para que se puedan ejecutar las pruebas, mismas que podrán ser modificadas según las razones de cada organización.

A continuación se señalan las condiciones mínimas que se deben presentar para iniciar la ejecución de las pruebas:

- Se poseen los escenarios de pruebas aprobados.
- El entorno de pruebas es el adecuado para el tipo de pruebas a iniciar.
- Los materiales y herramientas se encuentran disponibles.
- Se recibió la versión del software para pruebas con su correspondiente ambiente.
- Todos los recursos humanos y técnicos necesarios se encuentran disponibles.

Suspensión de pruebas

Así como se requieren algunas características para poder llevar a cabo la ejecución de las pruebas, se debe especificar las razones por las cuales se suspenderá el proceso hasta el cumplimiento de estas. Para ello se encuentra dicha sección, donde se detallarán todos los aspectos necesarios según el área responsable.

A continuación se describen los criterios por los cuales se podrá suspender la ejecución de las pruebas:

- No disposición del ambiente para verificación y validación.
- No contar los escenarios de pruebas.
- No contar con los requerimientos necesarios para la ejecución del servicio (Software, Hardware, documentación etc.).
- No contar con la versión a probar del sistema.
- Indisponibilidad de herramientas para la ejecución del servicio.

- No contar con todos los recursos humanos y técnicos necesarios.

Reanudación de pruebas

Una vez descritos los criterios de aceptación, en este apartado se describen las condiciones con las que se reanudarían las pruebas en caso de haber algún inconveniente o se presentara el punto anterior.

A continuación se indican los criterios por los cuales se podrá reanudar la ejecución de las pruebas:

- Disponibilidad de ambiente para verificación y validación.
- Contar con los requerimientos necesarios para la ejecución del servicio (Software, Hardware, documentación etc.).
- Contar con la versión a probar del sistema.
- Disponibilidad de herramientas para la ejecución del servicio.
- Todos los recursos humanos y técnicos necesarios se encuentran disponibles.

Validación y aceptación

Como todo software, este debe tener y cumplir con ciertas características deseables para poder ser liberado a producción. En este apartado se describen las características que debe tener dicha aplicación para tener calidad y funcionalidad, y con ello poder ser aprobado para su entrega.

A continuación se indican los criterios por los cuales se podrá dar por satisfactoria la validación de Calidad:

- Que el número de defectos sea menor al 1% por módulo, que su severidad no sea crítica, alta o media, y que estos no afecten el desempeño del sistema para continuar con el siguiente tipo de prueba.
- Que el número final de defectos en suma de todos sus componentes no sea mayor al 2%, que su severidad no sea crítica, alta o media, y que estos no afecten el desempeño del sistema.

Solo en estos casos se podrá dar por validado un experimento y el sistema en general para la liberación por parte del área de pruebas.

Revisión de documentos

En este apartado se menciona el lugar que alojara toda la documentación para su revisión y validación. Dicha actividad será gestionada por el líder del proyecto o en su defecto la persona responsable del almacenamiento y administración de la información. Para ello puede utilizarse un servidor para almacenar todos aquellos entregables a revisar, o bien, utilizar herramientas o servicios en la nube para cumplir con este fin. Es importante aclarar que solo en caso de aplicar y según la naturaleza del proyecto se recomienda utilizar herramientas especializada para la gestión, caso contrario se recomienda utilizar métodos tradicionales en base a políticas de la empresa.

Comunicación

La comunicación es un factor importante en todo proyecto, aquí se describirán todos aquellos medios de comunicación, así como la evidencia de los mismo y la frecuencia de dicha actividad, si por alguna circunstancia no llegara a describirse y aclararse este punto, puede repercutir en los tiempos de entrega del producto final. Ver tabla 9 para su registro.

Actor 1 / Rol	Actor 2 / Rol	Frecuencia	Medio de comunicación	Evidencia

Tabla 9. Tabla de comunicación (Elaboración propia).

Reporte de hechos

En esta sección se mencionaran todas aquellas incidencias encontradas relacionadas con el proyecto, con la finalidad de tener una bitácora bien definida que conlleve a una buena toma de decisiones por parte de la organización y poder erradicar problemas futuros en base a un historial creado. Se recomienda utilizar la tabla 10 para su registro.

Id	Incidencia	Fecha	Responsable / Rol

Tabla 10. Reporte de incidencias (Elaboración propia).

Definiciones

En este apartado, se deben mencionar y describir todos aspectos que te ayudaran a disipar dudas referentes al proceso de pruebas, serán tantos como sean necesarios si así se requiere ya que esto dependerán del área en cuestión.

A continuación se definen algunas palabras importantes mencionadas en el capítulo 2 para la comprensión del plan de pruebas y que estarán documentadas en el mismo de la siguiente forma:

"Pruebas integrales: Las pruebas de integración verifican si los componentes o subsistemas en su conjunto interactúan correctamente a través de su interfaz, cubren la funcionalidad establecida, y se ajustan a los requisitos especificados en las verificaciones correspondientes.

Pruebas funcionales: Se prueban las funcionalidades del sistema con las relaciones a los módulos definidos.

Pruebas de seguridad: Las Pruebas de Seguridad pretenden medir la Confidencialidad, Integridad y Disponibilidad de los datos tratando de identificar amenazas y riesgos desde el uso o interface de usuario final.

Pruebas de caja negra: Se prueban las funcionalidades de cada módulo con las respectivas entradas de datos y salidas de información.

El plan de prueba: Describe todos los métodos que se utilizarán para verificar que el software satisface la especificación del producto y las necesidades del cliente. Incluye los objetivos de calidad, necesidades de recursos, cronograma, asignaciones, métodos, etc.

Casos de prueba: Lista los artefactos específicos que serán probados y describe los pasos detallados que serán seguidos para verificar el software.

Reporte de pruebas: Describen los problemas encontrados al ejecutar los casos de prueba.

Herramientas de pruebas y automatización: Documentación de las herramientas empleadas en el proceso de pruebas.

Métricas, estadísticas y resúmenes: Indican como ha sido el progreso del proceso de prueba.

Estrategia: Serie de acciones/actividades a realizar para establecer un marco de trabajo.

Verificación: Actividad para revisar si un entregable o producto es correcto porque cumple con la especificación de su diseño.

Producto: Documento que es requerido en cada una de las fases y/o documentación con la que debe cumplir un rol determinado.

Incidencia: Documento que es requerido en cada una de las fases y/o documentación con la que debe cumplir un rol determinado".

SECCIÓN IV: DATOS TÉCNICOS

Firmas

En este último apartado se mencionaran y firmaran todas aquellas personas responsables de áreas que involucra el proceso de pruebas. Con esta última actividad, se da por concluido el proceso de pruebas y se da paso a la firma de liberación y culminación de forma satisfactoria.

Esta actividad es de suma relevancia, debido a que si no llegase a cerrar este proceso, no podría liberarse el producto por parte del área de pruebas, provocando no tener un producto con las características deseadas por el usuario final.

En la tabla 11 y 12 se muestra el ejemplo de cómo se debe llevar a cabo esta actividad. Cabe señalar que no existe un límite de personas para la firma del documento plan de pruebas final, sin embargo se recomienda que sea al menos una persona por cada área involucrada en el proceso.

Por parte del área de Pruebas:

Cargo
[nombre]

Tabla 11. Firma de líder de pruebas (Elaboración propia).

Por parte del área de Desarrollo:

Cargo
[nombre]

Tabla 12. Firma del líder de desarrollo (Elaboración propia).

3.1.2 Construcción y ejecución de pruebas

El guiarse de un modelo o metodología para ejecutar pruebas de software, implica seguir paso a paso todas las etapas que esta marca, solo así se llegara al cumplimiento de los objetivos. Para ello es necesario que el probador pueda entender todas y cada una de las etapas a detalle, con la finalidad que la actividad fluya de forma adecuada.

Una vez que se tiene el cumplimiento de la primera fase, la cual implica tener las plantillas para los escenarios de pruebas, la identificación de los módulos que se examinaran, el plan de pruebas terminado y aprobado por el líder del área y del proyecto, es tiempo se pasar a la segunda fase.

La segunda fase del proceso, está centrada en la construcción y ejecución de pruebas, en esta etapa se necesitarán de los elementos descritos en el plan como, herramientas, plantillas, recursos humanos y la versión del software a probar, así como el ambiente donde se ejecutaran las pruebas y los datos que se utilizaran para dicha ejecución.

Todo este proceso se desglosa en tres subetapas para poder tener una mejor gestión y administración. En este paso es de suma importancia seguir paso a paso el plan de pruebas que fue generado en el punto anterior, ya que será quien regirá y gobernara todo el proceso a partir de este punto, así mismo dentro del documento mencionado,

es indispensable se consulte constantemente el cronograma de actividades con la finalidad de entregar en tiempo y forma todas las tareas que fueron definidas para poder liberar el software con los lineamientos adecuados y descritos al inicio del proyecto.

A continuación se describen las tres etapas de las que está compuesta la tarea de construcción y ejecución de pruebas, con la finalidad de llevar a cabo una correcta ejecución del proceso de pruebas.

3.1.2.1 Identificación, clasificación y eliminación de casos de prueba redundantes

El primer paso, es la identificación de los módulos que serán probados, con ello se pretende abarcar cada funcionalidad del sistema así como su totalidad, con la finalidad que se cumplan las especificaciones y requisitos que fueron planteados al inicio del proyecto.

Una vez que se han identificado todos los casos de prueba, se pasa a la clasificación de estos en pruebas funcionales, pruebas integrales, pruebas de carga, pruebas de rendimiento y pruebas de seguridad. Esta parte es muy importante ya que si se llega a clasificar de forma incorrecta un módulo, puede ser que se omita probar de forma correcta y la totalidad de la funcionalidad del sistema, provocando fallos futuros y anomalías en su funcionalidad. Para esta clasificación, se recomienda sea validada por el líder del proyecto y líder del área de forma que se asegure incluir todos los componentes de forma adecuada.

Toda la información debe quedar descrita en el plan de pruebas mencionado en la sección IV en el apartado de “Elementos a Probar”, del tema 3.1.1.3, se debe especificar el componente que se va a probar, el tipo de experimento que se le va a ejecutar, el resultado parcial y final después del seguimiento, donde se indicaran con

números y porcentajes dichas consecuencias y el cierre del proceso de pruebas determinando si el módulo fue o no aprobado. Es importante mencionar que las iteraciones en la ejecución de pruebas, serán tantas como sean necesarias, esto hasta finalizar el cierre de cada uno de los errores encontrados, a continuación se presenta la tabla 13 para este fin.

Primera Iteración:

Componentes a probar	Tipo de prueba a ejecutar	Bugs / Sugerencias / Total de escenarios	Aprobado

Tabla 13. Elementos a probar (Elaboración propia).

Es importante mencionar que el presente modelo solo abarca 5 tipos de pruebas diferentes mencionadas en el párrafo anterior, sin embargo, si por parte de la empresa u organización se requiriera aplicar experimentos adicionales, se pueden incluir en el proceso, tomando en cuenta que se deben incorporar estas a los documentos, tanto el plan de pruebas, como la matriz de escenarios y el cronograma de trabajo.

Como paso final de esta etapa es necesario identificar si algunos de los casos que se han clasificado son redundantes o se encuentras repetidos, no solo se deben verificar elementos reiterados en cada una de las clasificaciones, si no también verificar que cada componente fue clasificado de forma adecuada en los cinco diferentes tipos de prueba para evitar realizar trabajo doble o de forma incorrecta.

3.1.2.2 Creación de casos y escenarios de prueba al sistema

Una vez que se han identificado, clasificado y eliminado los casos de prueba redundantes, el siguiente paso es la creación de los escenarios de prueba, en base a la clasificación que se realizó en el paso anterior, en este punto se tendrá una platilla

común para los cinco tipos de experimentos que será utilizada con el nombre correspondiente a las pruebas funcionales, pruebas integrales, pruebas de carga, pruebas de rendimiento y las pruebas de seguridad (Ver apéndice B).

En la figura 16, se presenta la platilla creada en mediante una hoja de cálculo que se debe utilizar para este apartado.

Registro de revisiones				
versión	Fecha	Autor	Revisor	Observaciones

MATRIZ DE DATOS – ESCENARIO DE PRUEBA [FUNCIONAL, INTEGRAL, SEGURIDAD, CARGA, RENDIMIENTO]
[ACRÓNIMO DEL SISTEMA] - M/[No. De Módulo] – CU/[No. De Caso de Uso] - [Nombre del caso de prueba]

INFORMACIÓN DETALLADA	
Identificador del caso de prueba	O] - M/[No. De Módulo] – CU/[No. De Caso de Uso] - ECP/[No De caso de Prueba] - [Nombre del caso de prueba] - [Nombre del escenario
Ruta	Ruta correspondiente para la ejecución del escenario de prueba
Precondiciones	Precondiciones a cumplir antes de iniciar la ejecución de los escenarios de prueba
Versión del sistema a Probar	Versión del sistema a probar

ID	Acción	Datos de entrada	Tipo de Prueba	Resultado Esperado	Resultado Final	Corregido
[ACRÓNIMO] - M/[No. De Módulo] - CU/[No. De caso de uso] -ECP/[No. De Escenario de caso de prueba] - C/[No. De caso]	[Descripción de lo que se realizara en el escenario]	[Información que se utilizara en campos donde se requiera información]	[Optimista / No optimista]	[Descripción del resultado esperado al ejecutar el escenario de prueba]	[Correcto / Incorrecto	[SI, NO, NO APLICA] En caso de no aplicar, es importante una breve descripción

Figura 16. Matriz para registro de escenarios de prueba (Elaboración propia).

Como podemos observar en la figura 16, podemos encontrar elementos importantes que son necesarios definir para su completo entendimiento y uso.

- Registro de versiones: Contendrá información relacionada con la administración del documento, se especificarán fechas, autores del documento, persona que aprobó el mismo y observaciones si fueran necesarias.
- Matriz de datos: Se debe especificar el tipo de prueba que se va a documentar, pudiendo ser, funcional, integral, carga, rendimiento o de seguridad.
- No. De Módulo: Correspondiente a la clasificación de los casos de uso, requisitos o clasificación que se tenga del sistema en documentación anterior. En caso de no existir, el área de pruebas podrá proponer una nueva numeración.

- No. De Caso de Uso: En caso de existir documentación que especifique los casos de uso del sistema, estos deberán tener un identificador, mismo que servirán para mencionarlos en este apartado.
- Nombre del caso de prueba: Nombre que se utilizara para identificar el caso de prueba, podrá ser propuesto por el área de pruebas en caso de no existir en documentos previos.
- Ruta: Para ejecutar un escenario de prueba especifico, es necesario indicar cuál fue la ruta que siguió para ejecutar el experimento, con la finalidad de reducir tiempos y poder replicar en cualquier momento el escenario.
- Precondiciones: Un aspecto relevante en la ejecución de pruebas son las precondiciones, en este apartado debe especificarse cuáles serán las actividades previas a cumplir para ejecutar un escenario de pruebas. En caso de no cumplirse con alguna de ellas, no se podrán llevar acabo las validaciones al software.
- Versión del sistema aprobar: Toda aplicación debe tener una versión estable a la cual se le aplicaran las pruebas, es indispensable que el área de indique a cual se deberán hacer todos los experimentos necesarios con la finalidad de detectar errores.
- ID: Una vez que se tiene la información fundamental, se crearan los escenarios específicos con cada combinación necesaria para probar el sistema. Para ello se requiere tenga un identificador único para un seguimiento correcto.
- Acción: Se debe describir de forma concreta cual será la acción que se llevara a cabo en el escenario, si es necesario realizar más de un paso, deberá describirse y numerarse para evitar confusiones.
- Datos de estrada: Para toda acción se requiere información para ejecutar el escenario, así como se enumeraron los paso, deberán especificarse los nombres y los valores que se ocuparan en cada campo, con la finalidad de evitar confusiones y realizar las acciones de forma correcta.
- Tipo de prueba: Existen dos tipos de pruebas que se deben realizar en cada escenario, las optimistas y las no optimistas. Las pruebas optimistas son

aquellas en la que esperamos un resultado satisfactorio al ejecutarla, las pruebas no optimistas son aquellas en las que esperamos un error al aplicar una acción.

- Resultado esperado: debe describirse de manera concreta cuál será la consecuencia después de haber ejecutado el escenario, esto implica describir uno o más datos resultantes que arrojara la acción.
- Resultado final: Para este punto existen dos estados, correcto e incorrecto. El resultado correcto será solo si la prueba fue satisfactorio en su totalidad, caso contrario tendrá que colocarse incorrecto y será un error que los programadores tendrán que corregir para la liberación del software.
- Corregido: Finalmente la última columna de la tabla se utilizara y actualizara cuando el desarrollador haya notificado la corrección del error. El probador tendrá que volver a ejecutar el escenario para validar que fue corregido de forma adecuada, caso contrario se tendrá que notificar al desarrollador del resultado. Para este apartado existen tres estado, si, no, no aplica.

  En el caso de si, se aplica cuando el defecto fue solucionado, caso contrario se utilizara no. Para no aplica, se utilizara cuando el error que se registró no es un error, es decir, que la funcionalidad o resultado de la acción es correcta y no debe modificarse su funcionamiento, esto puede llegar a ocurrir por confusión por parte del probador al no tener bien definido y entendido los requisitos del sistema.

A continuación se describe la nomenclatura utilizada para el documento que contiene los escenarios de prueba, ver tabla 14.

Tipo de documento	Descripción
[ACRÓNIMO DEL SISTEMA]	Siglas que identifican el proyecto.
M/[No. De Módulo]	Número del módulo correspondiente a la clasificación del sistema.
CU/[No. De Caso de Uso]	Número del caso de uso correspondiente a documentos previos.
[Nombre del caso de prueba]	Se recomienda que el nombre a utilizar sea significativo al módulo que se está probando.
ECP/[No De caso de Prueba]	Se recomienda utilizar numeración consecutiva para futuras consultas.
[Nombre del escenario de prueba]	El nombre del escenario hace referencia a la operación específica que se va a realizar. Por ello es indispensable sea significativo el nombre que se ocupe.
C/[No. De caso]	Este número será consecutivo para los experimentos que se tengan en el escenario de prueba.

Tabla 14. Descripción de nomenclatura (Elaboración propia).

Esta fase es una de los más importantes en las pruebas, aquí será donde se generaran todas las posibles combinaciones de los campos del sistema para poder identificar alguna anomalía o mal funcionamiento.

Para ello es importante tomar en cuenta los siguientes aspectos en cuanto a campos a probar:

- Probar las longitudes máximas y mínimas que se establecieron en los requisitos y base de datos del sistema. Por ejemplo si un campo está determinado por

aceptar un mínimo de 2 caracteres y un máximo de 10, las pruebas abarcarían probar dicho campo con 1 carácter y con 11 caracteres para validar que en realidad esta longitud no la acepta.

- Probar con diferentes tipos de dato, es decir, números enteros, números decimales, números negativos, letras, caracteres especiales, etc., para verificar que no tenga problemas con el tipo de codificación con el que fue programado, algunos ejemplos de ello son los siguientes: *$<!””/¡¿#$%&()[].; 123457 1.23 -12.
- Otro dato importante, es probar líneas de lenguajes de programación con el que fue desarrollado, como comentarios, asignación de variables, saltos de línea, excepciones, entre otras. Esto para verificar que no afectan el funcionamiento del sistema cuando el código es almacenado en la base de datos y posterior en su consulta genere problemas de ejecución o muestre código en la plataforma.
- Probar todos aquellos campos que fueron definidos como obligatorios para determinar si dejando los campos vacíos marca advertencias o errores que lleven al no cumplimiento de lo establecido.
- Si existieran campos opcionales, validar al igual que los obligatorios si al dejarlos en blanco el sistema realiza la operación o marca alguna excepción.
- Los campos en los cuales se realizaran cálculos, introducir valores incorrectos, números exponenciales, o con signos negativos con la finalidad de evaluar el tipo de valor que acepta y que puede trabajar dicho campo.
- Verificar en campos donde pide cargar imágenes los diferentes formatos como, .png, .jpg, .gif, .nmp, .jpeg, .tif, .tiff, etc., así como el tamaño y resolución de las mismas que pudieran afectar el almacenamiento de este tipo de archivos.
- En caso de que se requieran adjuntar documentos se deberán validar extensiones como, .doc, .xslx, .xls, .pdf, .doctx, .dot, .rtf, .txt, .htm, .docm, .xml, .thmx o algún otro archivo con extensiones diferentes a las que pide, esto para validar que el campo en realidad acepta solo lo señalado.

- En caso de tener vínculos, es importante validar que apunte y direccione a la ruta correcta según lo descrito por el enlace, así como identificar la ausencia de alguno.
- Si el sistema permite descargar archivos como imágenes, documentos o carpetas, es importante verificar la legibilidad y que el documento completo se haya descargado, así como validar que en realidad descargue el documento que menciona.
- Si el sistema permite realizar impresión de documentos, es indispensable verificar que el documento resultante cumpla con lo que el sistema mostraba en su versión digital.
- Algunos portales web o aplicaciones tiene la posibilidad de imprimir el sitio, por lo cual si fuese un requisito, es necesario que el portal pueda imprimirse de forma correcta sin mostrar código o imágenes desfasadas según su ubicación original en la página.
- Para las pruebas de carga, verificar la cantidad de usuarios que pueden ingresar al sistema al mismo tiempo, esto con la finalidad de validar el número total o estimado que puede soportar el sistema.
- El tiempo de respuesta es un factor relevante e importante, por lo que se debe validar que el sistema no tarde más de .10 o .3 segundos para dar respuesta, caso contrario puede resultar inoportuno para el usuario.
- Es importante que cada función y acción dentro del sistema tenga un éxito o fallo, por lo que se debe validar en primera instancia la acción y en segundo que el mensaje corresponda a la operación que se está realizando.

Es muy importante mencionar que para poder crear los escenarios, se debe tomar en cuenta poner un identificador a cada caso de uso, así como al documento que los contendrá, con ello se asegura que si en algún momento se requiere dar un seguimiento a un escenario o ver sobre que requisito repercute, este sea mapeado de forma fácil y sencilla y con ello evitar confusiones.

Finalmente todo documento debe tener un nombre para su identificación y seguimiento, a continuación se realiza una propuesta para los escenarios de prueba.

(ACRÓNIMO)-(MÓDULO)-(TIPO DE PRUEBA) - (NOMBRE DEL CASO DE PRUEBA)

Donde:

- Acrónico: Hace referencia a las siglas que lo identifican, ya sea la primera letra del nombre completo de la aplicación o bien algún calificativo según su función, esto dependerá de los primeros documentos creados donde vendrá descrito ese aspecto, caso contrario el área de pruebas tendrá que gestionar dicho acrónimo para su uso.
- Módulo: Este número debe ser tomado de documentos previos para llevar un seguimiento adecuado, y dependerá de cómo este dividido el proyecto, por lo que tendrá que ser consecutivo para cada documento nuevo.
- Tipo de prueba: Para identificar el tipo de prueba que se está aplicando a cada módulo, es necesario escribirlo en el nombre del documento, este puede ser, prueba funcional, integral, de seguridad, de carga y rendimiento.
- Nombre del caso de prueba: Hace referencia al módulo o funcionalidad del sistema que se está probando, importante utilizar nombres significativos.

Con el cumplimiento de todo lo descrito en este apartado se dará por concluida la fase y se podrá pasar a la siguiente actividad del proceso.

3.1.2.3 Ejecución de casos y escenarios de prueba al sistema

Para este apartado se presentan dos formas de cómo realizar la ejecución y reporte de los errores encontrados, la primera de ellas es mediante el documento de escenarios donde se registraran los resultados obtenidos en la ejecución de pruebas,

la segunda opción es mediante la herramienta *bugzilla*. A continuación se presentaran estos dos procesos, la elección de uno u otro dependerá del área, la empresa y la naturaleza del proyecto.

El primer método para generar y ejecutar las pruebas es con el apoyo del documento para escenarios de prueba. En esta actividad es indispensable que los probadores sigan al pie de la letra los escenarios que se han generado con anterioridad para poder comprobarlos en base a la información proporcionada, si la funcionalidad del sistema, así como las precondiciones, postcondiciones y rutas que se definieron son correctos, se dará por culminada la primera iteración. Solo así se tendrá un control y buena ejecución de los casos disponibles.

Como se describió en la figura 13 el en tema 3.1.2.1, una vez que se han ejecutado las pruebas, es necesario indicar el resultado de las mismas mediante los indicativos, correcto o incorrecto. Una vez que se realizó esta operación se debe obtener los resultados preliminares o de la primera interacción. Para ello se debe contabilizar la cantidad de errores encontrados, el número de sugerencias realizadas en determinado caso de pruebas y los aciertos en la aplicación, esto por cada tipo de prueba que se ejecutó. Posteriormente se debe notificar al desarrollador mediante el medio que se haya definido en el plan de pruebas, por ejemplo, vía correo electrónico, mediante una carpeta en la nube, o tradicionalmente por medio de dispositivos de almacenamiento como discos duros o USB, con la finalidad que el área técnica pueda atender dichas fallas.

Una vez que se tiene la información recabada, se debe plasmar en el plan de pruebas en el apartado "Elementos a probar" que se describió en la tabla 13 de este documento correspondiente a la primera iteración. Si en esta primera ejecución el 100 % de los experimentos fueron satisfactorios, no es necesario volver a ejecutar los ensayos. Sin embargo, si existieran errores en uno o más casos de pruebas, será necesario notificar el programador y volver a ejecutar las pruebas que ya fueron solucionadas para

confirmar dicha corrección, lo que dará paso al tema siguiente para el seguimiento de errores y resultados finales.

Un segundo procedimiento para la ejecución de pruebas es mediante la herramienta *bugzilla,* la cual nos permite realizar un registro de los escenarios que se han ido ejecutando, estos quedaran asentados en el servidor que se configuro al inicio de del proyecto, el cual a su vez mandara un mensaje de notificación al desarrollador indicando que hubo un error en el sistema que debe corregirse. Cabe señalar que toda la configuración y espacio de trabajo de la herramienta debe estar a cargo de una persona, con la finalidad de dar soporte y administrar la información.

Una vez que se mandó de forma automática el correo al desarrollador, este debe atender cada uno de los errores notificados por la herramienta y presentes en el software que se está desarrollando. Cuando las incidencias fueron atendidas, mediante la herramienta tendrá que notificar la atención a los fallos encontrados para que pueda ser informado al probador y validar la solución de estos. Si en la primera verificación, todos lo resultados fueron correctos y no se encontraron errores o sugerencias, el software se da por validado y se pasara a los resultados finales. De igual forma que en el primer proceso los resultados se deben plasmar en el plan de pruebas en el apartado "Elementos a probar" que se describió en la tabla 13 de este documento correspondiente a la primera iteración.

Es importante mencionar que si en la ejecución surgiera un caso el cual no fue contemplado en la creación de los escenarios, este debe documentarse y ejecutarse para no omitir pruebas que pudieran dejar errores en el sistema.

Para la ejecución de los escenarios es indispensable guiarse del plan de pruebas y verificar que los tiempos, las versiones y los criterios de ejecución se cumplen, para poder aplicar cada prueba de forma adecuada.

3.1.3 Seguimiento y resultados

La última etapa es crucial no solo para el seguimiento de incidencias encontradas, sino también, para presentar el resultado de las primeras pruebas al equipo de desarrollo para que estos puedan realizar las correcciones pertinentes y posteriormente poder validarlas y general el reporte final. Todo el proceso de seguimiento y resultados en cuanto a fechas, ciclo de vida de los bugs, reporte de incidencias encontradas y reporte final, está especificado en el plan de pruebas y que será adoptado y modificado por cada empresa según su conveniencia o necesidad.

Dado dos métodos para la ejecución de escenarios de prueba, recíprocamente existen dos métodos para el seguimiento y presentación de resultados. El primer método utilizando el documento de escenarios de prueba y el segundo método mediante la herramienta *bugzilla*. Es necesario precisar que si en la etapa anterior, “Ejecución de casos y escenarios de prueba al sistema” se utilizó la herramienta, para este paso debe seguirse la acción con la misma debido a la incompatibilidad entre esta última y el documento de escenarios de prueba. De igual forma si se utilizó el documento o plantilla base, se debe utilizar el mismo hasta el reporte de resultados finales y el cierre del proyecto.

Es importante indicar, si por alguna circunstancia se requiriera trabajar con la plantilla de escenarios y posteriormente utilizar la herramienta *bugzilla*, provocara problemas en los tiempos de entrega por el doble trabajo que se tiene que realizar en la documentación a entregar, así como, confusión al equipo técnico por el cambio en la presentación de errores en la ejecución escenarios al sistema.

A continuación se presentan las tres subfases de las que está compuesta la fase de seguimiento y resultados. Así mismo se describirán las herramientas y procesos utilizados para el cumplimiento de los objetivos.

3.1.3.1 Reporte y seguimiento de incidencias encontradas en la ejecución

Una vez que se ejecutaron las pruebas y se tienen las incidencias encontradas, deben ser reportadas a los programadores para su corrección. Cuando se haya cumplido por parte de los desarrolladores la tarea de verificar que en realidad sea un error y corregirlo, el probador confirma y valida que las correcciones realizadas por el programador son correctas para poder realizar el cierre de cada uno de ellos y aprobar el componente.

Un punto muy importante en el reporte y seguimiento de las incidencias es el plan de comunicación. Para ello se recomienda utilizar la herramienta antes mencionada que ayudará al seguimiento de incidencias mediante la comunicación entre probadores, programadores y la generación de reportes.

A continuación de describe la herramienta *Bugzilla* (Bugzilla et al., 2015):

Bugzilla permite organizar en múltiples formas los defectos de software, permitiendo el seguimiento de múltiples productos con diferentes versiones, a su vez compuestos de múltiples componentes. Permite además categorizar los defectos de software de acuerdo a su prioridad y severidad, así como asignarles versiones para su solución.

También permite anexar comentarios, propuestas de solución, designar a responsables a los que asignar la resolución y el tipo de solución que se aplicó al defecto, todo ello llevando un seguimiento de fechas en las cuales sucede cada evento, enviando mensajes de correo a los interesados en el error.

Bugzilla utiliza un servidor HTTP (como puede ser Apache) y una base de datos (normalmente, MySQL) para llevar a cabo su trabajo. Los errores pueden ser enviados por cualquier probador y pueden ser asignados a un desarrollador en particular. Cada falla puede tener diferente prioridad y encontrarse en diferentes estados, así como ir acompañado de notas del usuario o ejemplos de código que ayuden a corregir el error.

Posibles usos:

- Administración de Sistemas.
- Gestión de despliegue.
- Diseño de chips y problema de desarrollo de seguimiento (tanto pre y post fabricación).
- Seguimiento de software y hardware de error.
- Tecnologías de la información, colas de apoyo.

Con dicha herramienta los reportes y seguimiento de las incidencias encontradas en la ejecución se generan de forma automática debido a la robustez con la que cuenta *Bugzilla*, por ello el proceso de seguimiento y reporte se realiza tanto con los escenarios de prueba creados como con el instrumento disponible para el registro de fallos.

El primer proceso para el seguimiento con ayuda de la herramienta ejecutado el escenario que se documentó con anterioridad, si se llega a encontrar una funcionalidad incorrecta se registra en *Bugzilla* y este a su vez de forma automática manda correo al desarrollador que corresponda según sea el módulo configurado previamente con responsabilidad a una persona en específico. Una vez realizado esto el programador tendrá que abrir su cuenta en dicha plataforma para darle el seguimiento y corrección al bug registrado, este último indicara si aplica o no aplica su solución para que a su vez pueda ser informada al probador.

Finalmente una vez que se le notificó a la persona que realizo la prueba que esta ha sido solucionada, el probador tendrá que volver a ejecutar la prueba para confirmar que ya está remediada la falta, en caso satisfactorio en la plataforma se indicara que se solucionó de forma adecuada y se cerrara dicha anomalía, caso contrario el ciclo se repetirá notificando nuevamente al programador. Este proceso se deberá llevar a cabo tantas veces sea necesario hasta la solución total de las incidencias encontradas en el producto.

Sin embargo como se describió en la descripción, la herramienta requiere de una infraestructura especial para poder funcionar de forma correcta, por ello si no se tuviera

la posibilidad para poder utilizar *Bugzilla*, se recomienda que el seguimiento se lleve a cabo mediante el documento de escenarios de prueba.

Para el seguimiento en base al documento de escenarios de prueba, tendrá que realizarse vía correo electrónico o por algún otro medio que se elija para notificar al desarrollador de los fallos encontrados. Dentro del documentos de escenarios de prueba se colocara en su campo “Resultado final” si fue correcto o incorrecto. Con ello el programador tendrá la obligación actualizar el campo de “”Corregido” mencionado y descrito en la figura 2.14 con la leyenda, si, no, no aplica. Posteriormente tendrá que notificarle al probador mediante el mismo medio de la solución realizada, el verificador ejecutara el escenario y confirmara que el acierto fue correcto dando por solucionado y cerrado el escenario de prueba, caso contrario se repite el ciclo tantas veces sea necesario hasta su solución. Es importante recordar que por cada nueva validación que se realice al software, se genera un reporte correspondiente al número de iteración realizada, el resultado de esto debe ser plasmado en el pan de pruebas en la sección cuatro, apartado 4.1 elementos a probar.

3.1.3.2 Cierre y resultados finales de incidencias encontradas

Para este proceso existen dos formas de cerrar las incidencias y presentar los resultados finales de los experimentos del software, mediante la generación automática de reportes con la herramienta *bugzilla,* y por medio el documento de escenarios de prueba. En seguida se describirán ambos casos para su correcto uso y entendimiento.

Para el primer caso en el cierre de errores y resultado final de incidencias encontradas es mediante de la herramienta *Bugzilla*, una vez que se solucionaron todos los errores registrados y por parte del probador están validados y verificados, se debe generar el reporte final. Dicha herramienta tiene un apartado de reportes el cual se puede

configurarse con la información que se necesite, utilizando campos como, un identificador, tipo de prueba, tipo de componente que se probó, fecha de registro, fecha de solución, versión del sistema, pasos a seguir en la ejecución, estatus de un error, precondiciones, involucrados de las áreas, entre otros aspectos, para mayor información se puede consultar la página oficial de dicho instrumento.

Una vez que se tienen los campos establecidos para el reporte se procederá a generarlo. El resultado de esto será un documento con extensión .csv o xml, el cual podrá ser manipulado para cambiar la extensión y pasarlo a la extensión deseable para su mejor gestión. En dicho reportes se podrán observar todos aquellos errores que fueron solucionados, así como los que quedaron pendientes. Para el caso de fallos no solucionados, estos deberán de ser menor al 1 % por módulo y no mayor al 2 % del total, que su severidad, no sea, critica, alta o media, y que estos no afecten el desempeño del sistema.

Para el segundo método en el cierre y generación de resultados finales, basta con hacer el conteo en cada documento de escenarios de prueba, indicando la cantidad de errores encontrados y solucionados. Así mismo el porcentaje de errores permitidos para la liberación de algún producto de software, será con los mismos números mencionados en el párrafo pasado y con las mismas características en la criticidad de los fallos.

3.1.3.3 Generación de reporte final y análisis

Finalmente para concluir con todo el proceso de pruebas en indispensables general el reporte final para un análisis de posibles áreas de oportunidad a mejorar en algún proceso, tanto en la programación como en la ejecución de las pruebas.

Para ello basta con actualizar el documento “plan de pruebas”, en el apartado de “Elementos a probar” en su columna “Aprobado” tendrá que colocarse la leyenda SI para aprobar el módulo o NO para dar por hecho que dicho software no fue liberado

en base al resultado de las pruebas obtenidas. Es importante mencionar que el resultado final será posterior a la ejecución de un número indeterminado de verificaciones y validaciones sobre cada componente, por ello es indispensable guiarse de las fechas establecidas en el cronograma de trabajo y evitar con ello el desfase en la entrega del producto final.

Finalmente una vez que se ha actualizado el apartado de elementos a probar, se procederá a la firma de integrantes y representantes de cada una de las áreas involucradas, esto da por hecho que se liberó de forma satisfactoria el software y está listo para entregarse al usuario final, se recomienda también plasmar el autógrafo de conformidad por parte del cliente, con la finalidad de tener la evidencia de que acepto el producto final.

Con este paso se da por concluido todo el proceso de pruebas, dando por hecho que se libera un producto con las características y funcionalidades deseadas, así mismo cumpliendo los objetivos establecidos por el proyecto al inicio de este.

Finalmente como podemos ver, el modelo propuesto cuenta con un proceso bien definido, herramientas y plantillas para poder ser adoptado por cualquier empresa, organización o grupo de desarrollo de software con la finalidad de poder liberar productos de calidad al mercado y evitar costos en mantenimiento una vez lanzado el producto al mercado. Así mismo cuenta con definición y ejemplos de cada actividad y archivo presentado para un entendimiento y uso correcto para evitar malas interpretaciones del proceso en general.

A continuación se presenta la aplicación de la metodología a dos proyectos de software, con la finalidad de verificar el grado de eficacia en la elaboración, ejecución, descubrimiento y seguimiento de errores en el software. Cabe señalar que la información presentada dada la restricción del proyecto, no se mostrará datos específicos, para no repercutir en aspectos de divulgación de la aplicación próxima a liberar al mercado.

Capítulo 4 Aplicación del modelo propuesto

En base a todo lo expuesto en el capítulo anterior, en este apartado se describe y mencionan dos proyectos al cual le fue aplicada la metodología propuesta, dichos proyectos son software libre y de apoyo a la aplicación de métodos en el tema de lógica difusa, mismos que estarán disponibles al público en general según fechas determinadas por la organización creadora del sistema. Tal estudio consiste en verificar y validar la funcionalidad de estos dos proyectos antes de poder ser lanzados al mercado, si bien al ser de software libre y permitir que alguna empresa, organización o persona pueda contribuir al funcionamiento del programa, es indispensable que este pueda tener el mínimo de errores posibles para su correcto uso. En este capítulo se redacta el proceso de la metodología aplicada a ambos programas.

4.1 Composición de relaciones

El proyecto denominado “Composición de relaciones” como lo menciona el título e identificado por las siglas CR, tiene como objetivo primordial, ser una herramienta capaz de realizar operaciones con conjuntos difusos, involucrando el uso de operadores como unión, intersección y complemento, con la finalidad de proveer información que apoyara en la toma de decisiones en esta área.

Cabe mencionar, dada la restricción que se tiene del proyecto, no es posible documentar toda la información que compete a la aplicación, con la finalidad de no divulgar información indispensable del producto, por lo que, los escenarios de prueba y plan de pruebas, así como los resultados de los módulos no están especificados en este documento. Sin embargo se mostrará de forma general como se ejecutó cada una de las etapas y los resultados generales de esto.

A continuación se describen todas las fases del modelo y como fueron aplicadas a dicho proyecto.

4.1.1 Diseño de pruebas

4.1.1.1 Gestión de plantillas para pruebas

Como se describió en el capítulo 3, apartado 3.1.1.1 “Gestión de plantillas para pruebas”, la actividad principal es solicitar y gestionar con el área que lleve la administración y gestión de archivos, la documentación que se requerirá para el desarrollo de las actividades, es decir, el plan de pruebas y el documento para la creación de los escenarios de pruebas, así como el cronograma de actividades y el ciclo de vida de errores. Es importante mencionar que dicha área será o tendrá el nombre según las políticas de la empresa, en la mayoría de los casos denominada con el nombre “área de pruebas o área de calidad”, así mismo tendrá sus mecanismos y lugares de almacenamiento para un correcto control.

En el caso particular de este proyecto, no se requiere realizar alguna gestión adicional a alguna área dado que ya se cuenta con la documentación necesaria para la aplicación del modelo propuesto. Dentro de los documentos ocupados para la tarea esta, el plan de pruebas que regirá todo el proceso, la plantilla para generación y documentación de escenarios, el ejemplo del cronograma de actividades que guiara el proceso mediante fechas establecidas y el ciclo de vida de errores para indicar cuales son los pasos a seguir en el reporte y cierre de una falla. Esta documentación será suficiente para poder crear, ejecutar, dar seguimiento y cierre al proceso y errores encontrados en el software mencionado.

Por lo que, una vez cumplido con este punto se procedió al siguiente paso del modelo para seguir con lo establecido.

4.1.1.2 Identificación de requerimientos y casos de uso del sistema

El siguiente paso del modelo es la identificación de las necesidades del proyecto, para ello se utilizó el documento de requisitos que indica las características que debe tener y cumplir el software, donde se describe la acción y funcionamiento de cada componente, esto con la finalidad de identificar los posibles módulos a los que se aplicaran los tipos de pruebas.

A continuación se describen los requisitos generales del sistema:

Módulo 1: Configuración inicial

Soportar y permitir al usuario realizar la adición, eliminación y definición de conjuntos difusos, así como del universo de discurso, unidad de medida y la representación gráfica de los conjuntos difusos por parte de la interfaz, habilitando la selección de las 17 funciones de pertenencia tipo-1.

Módulo 2: Ingreso de valores y cálculo de grados de pertenencia

Soportar y permitir al usuario seleccionar una variable lingüística previamente definida y realizar el ingreso de valores para el cálculo de grados de pertenencia con respecto a los conjuntos difusos previamente definidos correspondientes a la variable lingüística seleccionada.

Módulo 3: Operaciones con conjuntos difusos

Soportar y permitir al usuario realizar el diseño de operaciones entre los conjuntos difusos previamente definidos por el mismo, soportando el uso de los elementos del triple de Morgan (t-norms, s-conorms y el complemento estándar) para realizar las operaciones de intersección, unión y complemento de conjuntos difusos.

Los requisitos descritos anteriormente, se analizaron para poder generar los casos de prueba, cabe mencionar que dicha clasificación no indica que de la misma forma sean numerados los escenarios, esto dependerá de la cantidad de casos que sean identificados por el probador para el cumplimiento de las pruebas de forma satisfactoria. En este caso fueron identificados 5 escenarios a los que se les ejecutaran los experimentos según la clasificación que se genere posteriormente.

4.1.1.3 Generación del plan de pruebas

La tercer actividad dentro del proceso de pruebas es la generación del plan que gobernara toda la etapa de pruebas, en este se mencionan todos los puntos que se describieron en el capítulo 3. Cabe mencionar que este proyecto ya fue concluido y liberado después de su evaluación de forma satisfactoria, por lo que dicho documento por su extensión y permisos de divulgación no es posible agregarlo en este apartado, sin embargo a continuación se presenta información general respecto al mismo.

En la tabla 15 y 16, se observa información general del documento plan de pruebas, donde se puede observar las fechas, versiones y creación de archivo, así como los participantes en estas dos actividades.

Realizado por	**Said Pérez Flores**	**Fecha**	**19/04/2016**
Aprobado por	Sergio Carlos Ponce Flores	Fecha	20/04/2016

Tabla 15. Revisiones al documento Plan de Pruebas (Elaboración propia).

Fecha	Versiones	Descripción	Autor
19/04/2016	1.0	Creación y configuración de archivo	Said Pérez Flores
20/04/2016	2.1	Aprobación y cierre del documento	Said Pérez Flores

Tabla 16. Versiones del documento Plan de pruebas (Elaboración propia).

En base a lo mencionado y descrito en el capítulo 3 de esta investigación, apartado 3.1.1.3 “Generación del plan de pruebas” el nombre de dicho sistema quedo de la siguiente forma:

CR - Plan de Pruebas

Donde las primeras dos letras hacen referencia al acrónimo del proyecto, seguido por el nombre genérico del archivo.

4.1.2 Construcción y ejecución de pruebas

4.1.2.1 Identificación, clasificación y eliminación de casos de prueba redundantes

Para este punto se generó una lista de los componentes a ser probados, se tomó en cuenta los requisitos del sistema, así como los módulos de los que estará conformado el sistema para poder obtener dicha información. Finalmente se clasificaron en dos grupos, pruebas funcionales y pruebas integrales con la finalidad de cubrir todos los componentes de una forma adecuada, una vez realizado esto se verifico que no existieran casos de prueba redundantes en la misma clasificación, para evitar perder el tiempo en ese aspecto. Si bien en el apartado 3.1.1.3 se mencionan 5 tipos de pruebas de las que está compuesto el modelo, para el caso de particular de la

aplicación solo se ocuparan las pruebas mencionadas ya que no existen los elementos necesarios para poder ejecutar pruebas de seguridad, rendimiento y carga.

A continuación se presentan los componentes a probar y su clasificación.

Componentes a probar	Tipo de prueba a ejecutar
Definición de universos	Funcional
Definición de relaciones	Funcional
Operaciones básicas con relaciones binarias	Integral
Composición de relaciones	Integral
Complementarias	Integral

Tabla 17. Componentes a probar (Elaboración propia).

4.1.2.2 Creación de casos y escenarios de prueba al sistema

Una vez identificados y clasificados los casos de prueba en el punto anterior, se pasa a la creación de escenarios. Es muy importante tomar en cuenta aspectos descritos en el capítulo 3, apartado 3.1.2.2 con el mismo título que este. En dicha sección se describen los puntos y aspectos que deben ser utilizados en la creación de un escenario, como longitud de campo, tipo de campo, campos obligatorios, tipo de archivos, entre otros. Una vez realizada la combinación pertinente de los escenarios de pruebas, se procedió a pasar al segundo punto en el proceso.

A continuación se describen los casos de prueba utilizados para la ejecución de los experimentos a la aplicación y la nomenclatura para el nombrado de archivos.

Nombre del documento	Nombre del caso de prueba
CR - 01 - Pruebas_Funcionales - Definición de universos	CR - M/01 - CU/00 - DEFINICIÓN DE UNIVERSOS
CR - 02 - Pruebas_Integrales - Definición de relaciones	CR - M/02 - CU/00 - DEFINICIÓN DE RELACIONES
CR - 03 - Pruebas_Integrales - Operaciones básicas	CR - M/03 - CU/00 - OPERACIONES BÁSICAS
CR - 04 - Pruebas_Integrales - Composición de relaciones	CR - M/04 - CU/00 - COMPOSICIÓN DE RELACIONES
CR - 05 - Pruebas_Funcionales – Complementarias	CR - M/05 - CU/00 - COMPLEMENTARIAS

Tabla 18. Descripción de casos de prueba (Elaboración propia).

Dichos nombres fueron escritos en base a lo especificado en el capítulo 3, apartado "Creación de casos y escenarios de prueba al sistema".

4.1.2.3 Ejecución de casos y escenarios de prueba al sistema

Una vez que se ha cumplido con la creación de los escenarios, el siguiente paso es ejecutar las pruebas funcionales e integrales a cada uno de los módulos existentes en el sistema. Para ello se utilizaron los documentos y casos de pruebas del punto anterior y se ejecutaron según las especificaciones descritas dentro de cada uno de ellos y con la información proporcionada por los mismos.

Es importante mencionar que para todos aquellos escenarios de prueba donde el experimento no fue satisfactorio, en el documento de escenarios de prueba, en el apartado resultado final, se colocó la leyenda "Incorrecto" y una breve descripción del

problema encontrado, así como la casilla de esta en rojo para su rápida identificación y corrección por la persona correspondiente, mientras que para los escenarios de prueba que son correctos se puso la leyenda “Correcto” con la finalidad de que estos sean omitidos por el programador que corrigió dicho módulo ver figura 17.

ID	Acción	Datos de entrada	Tipo de Prueba	Resultado Esperado	Resultado Final	Corregido
[ACRÓNIMO] – M/[No. De Módulo] – CU/[No. De caso de uso] –ECP/[No. De Escenario de caso de prueba] – C/[No. De caso]	[Descripción de lo que se realizara en el escenario]	[Información que se utilizara en campos donde se requiera información]	[Optimista / No optimista]	[Descripción del resultado esperado al ejecutar el escenario de prueba]	Incorrecto (Descripción del problema encontrado)	[SI, NO, NO APLICA] En caso de no aplicar, es importante una breve descripción
[ACRÓNIMO] – M/[No. De Módulo] – CU/[No. De caso de uso] –ECP/[No. De Escenario de caso de prueba] – C/[No. De caso]	[Descripción de lo que se realizara en el escenario]	[Información que se utilizara en campos donde se requiera información]	[Optimista / No optimista]	[Descripción del resultado esperado al ejecutar el escenario de prueba]	Correcto	[SI, NO, NO APLICA] En caso de no aplicar, es importante una breve descripción

Figura 17. Ejecución de escenarios de prueba (elaboración propia).

Posterior a la ejecución de las pruebas se procedió a notificar al desarrollador vía correo electrónico de los resultados obtenidos en las pruebas para su corrección. Es importante mencionar que dada la naturaleza y tamaño del proyecto, no se requirió ocupar la herramienta *Bugzilla* para el seguimiento de errores ya que no se contaba con los materiales y recursos necesarios para esto.

4.1.3 Seguimiento y resultados

4.1.3.1 Reporte y seguimiento de incidencias encontradas en la ejecución

Finalmente una vez que se han ejecutado y se tuvo el primer resultado de las pruebas, se generaron dos tipos de resultados, el primero dentro del documento de escenarios de prueba, en el cual se especifican como se indicó en el paso anterior todas las incidencias y sugerencias encontradas al ejecutar los escenarios de prueba. Posteriormente, mediante correo electrónico le fue notificado y mandado el archivo al desarrollador para que este pudiera corregir y verificar todas las fallas encontradas.

El segundo reporte de la primera iteración realizada al sistema fue descrita en el plan de pruebas, para consulta por el área administrativa y de desarrollo del proyecto, ayuda a tener una primera perspectiva de cómo va avanzando la validación de la aplicación ya que son datos generales y porcentuales los que se especifican. En ella se describe aspectos como, la cantidad de errores encontrados, las sugerencias que se realizan al programa y la cantidad de pruebas realizadas por módulo.

En las siguientes tablas pertenecientes al plan de pruebas, se presentan los resultados obtenidos de la primera, segunda y tercer iteración en la ejecución y validación de errores encontrados.

Resultados de la primera iteración:

Componentes a probar	Tipo de prueba a ejecutar	Bugs / Sugerencias / Total de escenarios	Aprobado
Definición de universos	Funcional	13 / 0 / 25	NO
Definición de relaciones	Funcional	16 / 0 / 29	NO
Operaciones básicas con relaciones binarias	Integral	2 / 8 / 18	NO
Composición de relaciones	Integral	0 / 0 / 28	SI
Complementarias	Integral	27 / 0 / 30	NO
	Total	58 / 8 / 130 44.6% / 6.1% / 100%	NO

Tabla 19. Resultado de las primeras pruebas (Elaboración propia).

Resultados de la segunda iteración de pruebas:

Componentes a probar	Tipo de prueba a ejecutar	Bugs / Sugerencias / Total de escenarios	Aprobado
Definición de universos	Funcional	0 / 0 / 25	SI
Definición de relaciones	Funcional	0 / 0 / 29	SI
Operaciones básicas con relaciones binarias	Integral	0 / 0 / 18	SI
Composición de relaciones	Integral	0 / 0 / 28	SI
Complementarias	Integral	27 / 0 / 30	NO
	Total	27 / 8 / 130 20.8 % / 6.1% / 100%	NO

Tabla 20. Resultados de la segunda iteración de pruebas (Elaboración propia).

En las tablas 19, 20 se presentan los resultados de las iteraciones realizadas al sistema, con las cuales una vez notificado al programador, este a su vez corrige y notifica al área de pruebas de su acción.

Por cada iteración se notifica al desarrollador de las incidencias encontradas, para que este a su vez pueda corregirlas y poder ser validadas por el área de pruebas.

Para poder notificar al desarrollador que un error lo corrigió de forma satisfactoria una vez validado por el área de pruebas, mediante el documento escenarios de prueba en el campo “resultado final” donde se colocó previamente la descripción del error y la casilla se puso en rojo, para este paso se cambió la casilla a verde y en el campo “corregido” se colocó la leyenda “SI”. Como se muestra en la siguiente imagen.

ID	Acción	Datos de entrada	Tipo de Prueba	Resultado Esperado	Resultado Final	Corregido
[ACRÓNIMO] – M/[No. De Módulo] – CU/[No. De caso de uso] –ECP/[No. De Escenario de caso de prueba] – C/[No. De caso]	[Descripción de lo que se realizara en el escenario]	[Información que se utilizara en campos donde se requiera información]	[Optimista / No optimista]	[Descripción del resultado esperado al ejecutar el escenario de prueba]	Incorrecto (Descripción del problema encontrado)	SI
[ACRÓNIMO] – M/[No. De Módulo] – CU/[No. De caso de uso] –ECP/[No. De Escenario de caso de prueba] – C/[No. De caso]	[Descripción de lo que se realizara en el escenario]	[Información que se utilizara en campos donde se requiera información]	[Optimista / No optimista]	[Descripción del resultado esperado al ejecutar el escenario de prueba]	Correcto	

Tabla 21. Seguimiento de errores (Elaboración propia).

Este paso se realiza tantas veces sea necesario con todos los errores que se corrijan e ira de la mano con cada reporte general de iteraciones. Todo concluye en el momento que en la última iteración no se tienen fallos a corregir y el área de pruebas ha verificado y validado cada una de las incidencias encontradas.

Este proceso se llevó acabo con tres iteraciones para poder obtener el 100% de las pruebas correctas, en una segunda iteración se logró reducir del total de 44.6% de errores a solo el 13% de errores aun presentes, finalmente con la tercera iteración del 13% que restaban por solucionar se logró corregir dicho porcentaje obteniendo un producto de calidad al lograr corregir el 100 por ciento de los errores detectados en las pruebas gracias el correcto seguimiento que se realizó en ellos.

4.1.3.2 Cierre y resultados finales de incidencias encontradas

Una vez concluido con todo lo anterior se realizó el reporte final de los elementos probados, cerrando el ciclo de pruebas por el área correspondiente, con ello queda liberado el producto para poder ser probado si así lo requiriera algún otro ente.

En base a los resultados obtenidos se puede concluir que el software cumple con el proceso de verificación y validación que el modelo propuesto detalla para liberar software de calidad al mercado, asegurando no solo la funcionalidad del sistema, sino también una buena navegación entre los componentes del mismo.

A continuación se presentan los resultados finales obtenidos de la tercera iteración del sistema.

Componentes a probar	Tipo de prueba a ejecutar	Bugs / Sugerencias / Total de escenarios	Aprobado
Definición de universos	Funcional	0 / 0 / 25	SI
Definición de relaciones	Funcional	0 / 0 / 29	SI
Operaciones básicas con relaciones binarias	Integral	0 / 0 / 18	SI
Composición de relaciones	Integral	0 / 0 / 28	SI
Complementarias	Integral	0 / 0 / 30	SI
	Total	0 / 0 / 130 0% / 0% / 100%	

Tabla 22. Resultado final de pruebas (Elaboración propia).

4.1.3.3 Generación de reporte final y análisis

Finalmente una vez concluido con el proceso de pruebas, esta última etapa hace referencia a la actualización del Plan de Pruebas donde estará toda la información referente a las pruebas que se realizaron a la aplicación, dicho escrito requerirá las firmas de las partes involucradas y así mismo este será entregado al líder del proyecto, en base a los resultados podrá analizar y deliberar si existiera alguna actividad que mejorar.

Con ello se dará por concluido el proceso de pruebas de software para dar paso a probar a otra aplicación. Es importante aclarar que dada la naturaleza de la aplicación no fue requerido el uso de la herramienta *Bugzilla*, por lo que toda la comunicación fue vía correo electrónico, y los documentos que fueron utilizados, fueron las plantillas descritas en el capítulo 3.

4.2 Operaciones con conjuntos difusos

El proyecto denominado “Operación con conjuntos difusos” como lo menciona el título e identificado por las siglas CR, tiene como objetivo primordial, ser una herramienta capaz de realizar operaciones con conjuntos difusos, involucrando el uso de operadores como unión, intersección y complemento, con la finalidad de proveer información que apoyara en la toma de decisiones en esta área.

Cabe mencionar, dada la restricción que se tiene del proyecto, no es posible documentar toda la información que compete a la aplicación, con la finalidad de no divulgar información indispensable del producto, por lo que, los escenarios de prueba y plan de pruebas, así como los resultados de los módulos no están especificados en este documento. Sin embargo se mostrará de forma general como se ejecutó cada una de las etapas y los resultados generales de esto.

A continuación se describen todas las fases del modelo y como fueron aplicadas a dicho proyecto

4.2.1 Diseño de pruebas

4.2.1.1 Gestión de plantillas para pruebas

Como se describió en el capítulo 3, apartado 3.1.1.1 “Gestión de plantillas para pruebas”, la actividad principal es solicitar y gestionar con el área que lleve la administración y gestión de archivos, la documentación que se requerirá para el desarrollo de las actividades, es decir, el plan de pruebas y el documento para la creación de los escenarios de pruebas, así como el cronograma de actividades y el ciclo de vida de errores. Es importante mencionar que dicha área será o tendrá el nombre según las políticas de la empresa, en la mayoría de los casos denominada con

el nombre “área de pruebas o área de calidad”, así mismo tendrá sus mecanismos y lugares de almacenamiento para un correcto control.

En el caso particular de este proyecto, no se requiere realizar alguna gestión adicional a alguna área dado que ya se cuenta con la documentación necesaria para la aplicación del modelo propuesto. Dentro de los documentos ocupados para la tarea esta, el plan de pruebas que regirá todo el proceso, la plantilla para generación y documentación de escenarios, el ejemplo del cronograma de actividades que guiara el proceso mediante fechas establecidas y el ciclo de vida de errores para indicar cuales son los pasos a seguir en el reporte y cierre de una falla. Esta documentación será suficiente para poder crear, ejecutar, dar seguimiento y cierre al proceso y errores encontrados en el software mencionado.

Por lo que, una vez cumplido con este punto se procedió al siguiente paso del modelo para seguir con lo establecido.

4.2.1.2 Identificación de requerimientos y casos de uso del sistema

Antes de poder seguir con el proceso de pruebas se utilizó el documento de requisitos que indica las características que debe tener y cumplir el software, donde se describe la acción y funcionamiento de cada componente.

A continuación se describen los requisitos del sistema:

Módulo 1: Definición de universos de discurso

Soportar y permitir al usuario realizar la adición, eliminación y definición de universos de discurso para la creación y definición de relaciones, así como mostrar al usuario la representación vectorial de dichos universos de discurso.

Módulo 2: Definición de relaciones

Soportar y permitir al usuario realizar la adición, eliminación y definición de relaciones entre universos de discurso, tanto binaria como difusa, para el cálculo de operaciones básicas binarias y para la composición de relaciones, así como mostrar al usuario la representación matricial de dichas relaciones.

Módulo 3: Operaciones básicas con relaciones binarias

Soportar la definición y el cálculo de operaciones básicas con relaciones binarias, utilizando los universos de discurso y las relaciones previamente definidas, soportando el uso de los elementos del triple de Morgan (t-norms, s-conorms y el complemento estándar) para realizar las operaciones de intersección, unión y complemento de conjuntos difusos.

Módulo 4: Composición de relaciones

Soportar la definición y el cálculo de las composiciones de relaciones diseñadas por el usuario, utilizando los universos de discurso y relaciones previamente definidas, soportando el uso de los elementos del triple de Morgan (t-norms, s-conorms y el complemento estándar) para realizar las operaciones de intersección, unión y complemento de conjuntos difusos.

Los requisitos descritos anteriormente, se analizaron para poder generar los casos de prueba, cabe mencionar que dicha clasificación no indica que de la misma forma sean numerados los escenarios, esto dependerá de la cantidad de casos que sean identificados por el probador para el cumplimiento de las pruebas de forma satisfactoria. En este caso fueron identificados 5 escenarios a los que se les ejecutaran los experimentos según la clasificación que se genere posteriormente.

4.2.1.3 Generación del plan de pruebas

La tercer actividad dentro del proceso de pruebas es la generación del plan que gobernara toda la etapa de pruebas, en este se mencionan todos los puntos que se describieron en el capítulo 3. Cabe mencionar que este proyecto ya fue concluido y liberado después de su evaluación de forma satisfactoria, por ello dicho documento por su extensión y restricción de información, no es posible agregarlo en este apartado, sin embargo a continuación se presenta información general respecto al mismo.

En la tabla 24 y 25, se observa información general del documento plan de pruebas, donde se puede observar las fechas, versiones y creación de archivo, así como los participantes en estas dos actividades.

Realizado por	**Said Pérez Flores**	**Fecha**	**27/04/2016**
Aprobado por	Sergio Carlos Ponce Flores	Fecha	28/04/2016

Tabla 23. Revisiones al documento Plan de Pruebas (Elaboración propia).

Fecha	Versiones	Descripción	Autor
27/04/2016	1.0	Creación y configuración de archivo	Said Pérez Flores
28/04/2016	2.1	Aprobación del documento	Said Pérez Flores

Tabla 24. Versiones del documento Plan de pruebas (Elaboración propia).

En base a lo mencionado y descrito en el capítulo 3 de esta investigación, apartado 3.1.1.3 “Generación del plan de pruebas” el nombre de dicho sistema quedo de la siguiente forma:

OPCD - Plan de Pruebas

Donde las primeras dos letras hacen referencia al acrónimo del proyecto, seguido por el nombre genérico del archivo.

4.2.2 Construcción y ejecución de pruebas

4.2.2.1 Identificación, clasificación y eliminación de casos de prueba redundantes

Para este punto se generó una lista de los componentes a ser probados, se tomó en cuenta los requisitos del sistema, así como los módulos de los que estará conformado el sistema para poder obtener dicha información. Finalmente se clasificaron en dos grupos, pruebas funcionales y pruebas integrales con la finalidad de cubrir todos los componentes de una forma adecuada, una vez realizado esto se verifico que no existieran casos de prueba redundantes en la misma clasificación, para evitar perder el tiempo en ese aspecto. Si bien en el apartado 3.1.1.3 se mencionan 5 tipos de pruebas de las que está compuesto el modelo, para el caso de particular de la aplicación solo se ocuparan las pruebas mencionadas ya que no existen los elementos necesarios para poder ejecutar pruebas de seguridad, rendimiento y carga..

A continuación se presentan los componentes probados y su clasificación:

Componentes a probar	Tipo de prueba a ejecutar
Configuración inicial	Funcional
Calculo de grados de pertenencia	Integral
Operaciones con conjuntos difusos	Integral
Complementarias	Integral
Configuración inicial	Funcional

Tabla 25. Componentes a probar (Elaboración propia).

4.2.2.2 Creación de casos y escenarios de prueba al sistema

Una vez identificados y clasificados los casos de prueba en el punto anterior, se pasa a la creación de escenarios. Es muy importante tomar en cuenta aspectos descritos en el capítulo 3, apartado 3.1.2.2 con el mismo título que este. En dicha sección se describen los puntos y aspectos que deben ser utilizados en la creación de un escenario, como longitud de campo, tipo de campo, campos obligatorios, tipo de archivos, entre otros. Una vez realizada la combinación pertinente de los escenarios de pruebas, se procedió a pasar al segundo punto en el proceso.

A continuación se describen los casos de uso utilizados para la ejecución de pruebas a la aplicación, así como la nomenclatura que debe utilizarse.

Nombre del documento	Nombre del caso de prueba
OPCD - 01 - Pruebas_Funcionales - Configuración inicial	OPCD - M/01 - CU/00 - CONFIGURACIÓN INICIAL
OPCD - 02 - Pruebas_IntegralesI - Calculo de grados de pertenencia	OPCD - M/02 - CU/00 - CALCULO DE GRADOS DE PERTENENCIA
OPCD - 03 - Pruebas_Integrales - Operaciones con conjuntos difusos	CR - M/03 - CU/00 - OPERACIONES BÁSICAS
OPCD - 04 - Pruebas_Funcionales - Complementarias	CR - M/04 - CU/00 - COMPLEMENTARIAS

Tabla 26. Descripción de casos de prueba (Elaboración propia).

Dichos nombres fueron escritos en base a lo especificado en el capítulo 3, apartado "Creación de casos y escenarios de prueba al sistema".

4.2.2.3 Ejecución de casos y escenarios de prueba al sistema

Una vez que se ha cumplido con la creación de los escenarios, el siguiente paso es ejecutar las pruebas funcionales e integrales a cada uno de los módulos existentes en el sistema. Para ello se utilizaron los documentos y casos de pruebas del punto anterior y se ejecutaron según las especificaciones descritas dentro de cada uno de ellos y con la información proporcionada por los mismos.

Es importante mencionar que para todos aquellos escenarios de prueba donde el experimento no fue satisfactorio, en el documento de escenarios de prueba, en el apartado resultado final, se colocó la leyenda “Incorrecto” y una breve descripción del problema encontrado, así como la casilla de esta en rojo para su rápida identificación y corrección por la persona correspondiente, mientras que para los escenarios de prueba que son correctos se puso la leyenda “Correcto” con la finalidad de que estos sean omitidos por el programador que corrigió dicho módulo ver figura 17.

Posterior a la ejecución de las pruebas se procedió a notificar al desarrollador vía correo electrónico de los resultados obtenidos en las pruebas para su corrección. Es importante mencionar que dada la naturaleza y tamaño del proyecto, no se requirió ocupar la herramienta *Bugzilla* para el seguimiento de errores ya que no se contaba con los materiales y recursos necesarios para esto.

4.2.3 Seguimiento y resultados

4.2.3.1 Reporte y seguimiento de incidencias encontradas en la ejecución

Finalmente una vez que se han ejecutado y se tuvo el primer resultado de las pruebas, se generaron dos tipos de resultados, el primero dentro del documento de escenarios de prueba, en el cual se especifican como se indicó en el paso anterior todas las incidencias y sugerencias encontradas al ejecutar los escenarios de prueba. Posteriormente, mediante correo electrónico le fue notificado y mandado el archivo al desarrollador para que este pudiera corregir y verificar todas las fallas encontradas.

El segundo reporte de la primera iteración realizada al sistema fue descrita en el plan de pruebas, para consulta por el área administrativa y de desarrollo del proyecto, ayuda a tener una primera perspectiva de cómo va avanzando la validación de la aplicación ya que son datos generales y porcentuales los que se especifican. En ella

se describe aspectos como, la cantidad de errores encontrados, las sugerencias que se realizan al programa y la cantidad de pruebas realizadas por módulo.

En las siguientes tablas pertenecientes al plan de pruebas, se presentan los resultados obtenidos de la primera, segunda y tercer iteración en la ejecución y validación de errores encontrados.

Resultados de la primera iteración:

Componentes a probar	Tipo de prueba a ejecutar	Bugs / Sugerencias / Total de escenarios	Aprobado
Configuración inicial	Funcional	9 / 1 / 33	NO
Calculo de grados de pertenencia	Integral	0 / 3 / 6	NO
Operaciones con conjuntos difusos	Integral	0 / 0 / 9	SI
Complementarias	Integral	15 / 0 / 30	NO
	Total	24 /4 / 78 31% / 5.1% / 100%	

Tabla 27. Resultado de las primeras pruebas (Elaboración propia).

Resultados de la segunda iteración de pruebas:

Componentes a probar	Tipo de prueba a ejecutar	Bugs / Sugerencias / Total de escenarios	Aprobado
Configuración inicial	Funcional	0 / 0 / 33	
Calculo de grados de pertenencia	Integral	0 / 0 / 6	
Operaciones con conjuntos difusos	Integral	0 / 0 / 9	
Complementarias	Integral	15 / 0 / 30	
	Total	15 /0 / 78 31% / 5.1% / 100%	

Tabla 28. Resultado de la segunda iteración de pruebas (Elaboración propia).

En las tablas 19, 20 se presentan los resultados de las iteraciones realizadas al sistema, con las cuales una vez notificado al programador, este a su vez corrige y notifica al área de pruebas de su acción.

Por cada iteración se notifica al desarrollador de las incidencias encontradas, para que este a su vez pueda corregirlas y poder ser validadas por el área de pruebas.

Para poder notificar al desarrollador que un error lo corrigió de forma satisfactoria una vez validado por el área de pruebas, mediante el documento escenarios de prueba en el campo “resultado final” donde se colocó previamente la descripción del error y la casilla se puso en rojo, para este paso se cambió la casilla a verde y en el campo “corregido” se colocó la leyenda “SI”. Como se muestra en la tabla 22.

Este proceso se llevó acabo con tres iteraciones para poder obtener el 100% de las pruebas correctas, en la cual en una segunda iteración se logró reducir del total de 31% de errores a solo el 19.3% aun presentes, finalmente con la tercera iteración del

19.3% que restaban por solucionar, se logró corregir dicho porcentaje obteniendo un producto de calidad al lograr corregir el 100 por ciento de los errores detectados en las pruebas gracias el correcto seguimiento que se realizó en ellos.

4.2.3.2 Cierre y resultados finales de incidencias encontradas

Una vez concluido con todo lo anterior se realizó el reporte final de los elementos probados, cerrando el ciclo de pruebas por el área correspondiente, con ello queda liberado el producto para poder ser probado si así lo requiriera algún otro ente.

En base a los resultados obtenidos se puede concluir que el software cumple con el proceso de verificación y validación que el modelo propuesto detalla para liberar software de calidad al mercado, asegurando no solo la funcionalidad del sistema, sino también una buena navegación entre los componentes del mismo.

A continuación se presentan los resultados finales obtenidos de la tercera iteración del sistema.

Componentes a probar	Tipo de prueba a ejecutar	Bugs / Sugerencias / Total de escenarios	Aprobado
Definición de universos	Funcional	0 / 0 / 25	SI
Definición de relaciones	Funcional	0 / 0 / 29	SI
Operaciones básicas con relaciones binarias	Integral	0 / 0 / 18	SI
Composición de relaciones	Integral	0 / 0 / 28	SI
Complementarias	Integral	0 / 0 / 30	SI
	Total	0 / 0 / 130 0% / 0% / 100%	

Tabla 29. Resultado final de pruebas (Elaboración propia).

4.2.3.3 Generación de reporte final y análisis

Finalmente, una vez concluido con el proceso de pruebas, esta última etapa hace referencia a la actualización del documento plan de pruebas donde está toda la información referente a los experimentos que se realizaron a la aplicación, dicho escrito requirió las firmas de las partes involucradas, así mismo este le fue entregado al líder del proyecto para que valide el proceso. En base a los resultados podrá analizar

y deliberar si existiera alguna actividad que mejorar, esta actividad es de relevancia ya que mediante las deducciones se pueden cambiar áreas de proceso, la forma en cómo se realiza una actividad o la inclusión de procesos para la mejora en el desarrollo de software.

Con ello se dará por concluido el proceso de pruebas de software para dar paso a probar a otra aplicación. Es importante aclarar que dada la naturaleza de la aplicación no fue requerido el uso de la herramienta *Bugzilla*, por lo que toda la comunicación fue vía correo electrónico, y los documentos que fueron utilizados, fueron las plantillas descritas en el capítulo 3.

Como se puede observar en la aplicación del modelo a dos proyectos, se presentó un proceso definido en todas sus etapas para la generación, ejecución, seguimiento y cierre de errores a dos sistemas de software, generando con ello no solo la liberación de un producto de calidad, sino también, la ejecución del proceso de forma fácil gracias a las plantillas que la metodología expone, así como la guía de cómo generar escenarios de prueba.

Es importante mencionar que la aplicación del presente modelo, estaba inicialmente planeado ser aplicado a 5 diferentes sistemas informáticos, entre ellos: software de escritorio, software para servicios web y software de aplicaciones móviles. Sin embargo, debido al desfase por parte del área de desarrollo en el tiempo de entrega por parte de estos, solo se pudo aplicar a 2 sistemas, mismos que fueron descritos en este capítulo.

Capítulo 5 Resultados y análisis

En este capítulo se describe el resultado de la evaluación de dos proyectos al cual le fue aplicada la metodología propuesta en esta investigación, los comentarios finales y análisis del proceso.

Así mismo, en base a los resultados obtenidos de estas dos aplicaciones se realiza una comparativa del comportamiento para validar le eficiencia y eficacia del proceso de pruebas versus las metodologías tradicionales y existentes, así como la correcta comunicación entre los involucrados para su correcto funcionamiento.

Finalmente, se presenta una comparativa entre el modelo propuesto y los existentes y descritos en el capítulo 1, con la finalidad de presentar información que lleve a la determinación en la mejora del proceso de pruebas y herramientas proporcionadas por estos, en base a los resultados obtenidos.

5.1 Resultados y comentarios finales de la aplicación

En el capítulo 4 se presentó la aplicación de la metodología a dos aplicaciones de software, con el nombre “Composición de relaciones” y “Operaciones con conjuntos difusos”, dando como resultados la liberación de estos sistemas de forma satisfactoria. Cabe señalar que el proceso de pruebas se llevó acabo de forma fácil y eficiente dando como resultado una aplicación de calidad.

Dentro de las plantillas y herramientas utilizadas en la aplicación del modelo fueron las siguientes:

- Plan de pruebas.
- Escenarios de pruebas.

- Cronograma de actividades.
- Ciclo de vida de errores.

Como se puede observar no se utilizó la herramienta *Bugzilla debido a que no se contaban con los materiales y personal necesarios para su configuración, por lo que todo el seguimiento se llevó a cabo mediante el documento de escenarios de prueba.*

Posterior al proceso de pruebas y después de tres iteraciones para el seguimiento de errores en cada uno de los proyectos, se generaron resultados satisfactorios que ayudaron a determinar la liberación de productos de calidad y que confirman la eficiencia y eficacia del modelo propuesto para evaluar software.

En la tabla 30 y 31 se presenta el concentrado final de las iteraciones en la ejecución y evaluación de las aplicaciones de software.

Composición de relaciones:

Iteración	Pruebas Total / Porcentaje	Correctos	Errores	Sugerencias
Primera	130 / 100 %	64 / 49.2 %	58 / 44.7 %	8 / 6.1 %
Segunda		95 / 73.1 %	27 / 20.8 %	8 / 6.1 %
Tercera		130 / 100 %	0 / 0 %	0 / 0 %

Tabla 30. Resultados finales (Elaboración propia).

En la tabla 30 se observan los resultados obtenidos de las tres iteraciones al proyecto, donde se realizó un total de 130 experimentos al software, de los cuales, en la primera ejecución de los escenarios se lograron detectar 58 errores equivalentes al 44.7% y 8 sugerencias equivalentes al 6.1%, realizando la suma de estos dos factores nos da un

total de 50.8 % de fallos encontrados del total de pruebas. Con ello podemos decir que si no se hubiese ejecutado la metodología y plan de pruebas, la aplicación pudo haber salido al mercado con dichos porcentajes, lo que hubiese provocado pérdida de tiempo, esfuerzo y dinero en el mantenimiento al mismo.

En base al modelo propuesto y ejecutado al software, en una segunda iteración, dando seguimiento a los defectos encontrados y gracias a la buena comunicación entre el área de pruebas y desarrollo, se logró disminuir el porcentaje en 27 errores equivalentes al 20.8%, en cuanto a las sugerencias dada la criticidad baja que tenían no se modificó o soluciono alguno.

Finalmente, en una tercera iteración, dando seguimiento mediante el procedimiento expuesto en el modelo de la presente investigación, se logró obtener el grado de satisfacción deseado, obteniendo un total de 0 errores y 0 sugerencias, gracias al proceso y plantillas ocupados de la metodología propuesta.

Con esto se dio por concluido y liberado el proceso de pruebas, logrando verificar y validad que el producto de software cumpla con las características y calidad deseada.

Operaciones con conjuntos difusos:

Iteración	Pruebas	Correctos	Errores	Sugerencias
Primera	78 / 100 %	50 / 64 %	24 / 31 %	4 / 5 %
Segunda		63 / 80.7 %	15 / 19.3 %	0 / 0 %
Tercera		78 / 100 %	0 / 0 %	0 / 0 %

Tabla 31. Resultados finales (Elaboración propia).

En la tabla 31 se observan los resultados obtenidos de las tres iteraciones al proyecto, donde se realizó un total de 78 experimentos al software, 52 ensayos menos que en

la primer aplicación, de los cuales, en la primera ejecución de los escenarios se lograron detectar 24 errores equivalentes al 31% y 4 sugerencias equivalentes al 5%, realizando la suma de estos dos factores nos da un total de 36% de fallos encontrados del total de pruebas. Con ello podemos decir que si no se hubiese ejecutado la metodología y plan de pruebas, la aplicación pudo haber salido al mercado con dichos porcentajes, lo que hubiese provocado pérdida de tiempo, esfuerzo y dinero en el mantenimiento al mismo.

En base al modelo propuesto y ejecutado al software, en una segunda iteración, dando seguimiento a los defectos encontrados y gracias a la buena comunicación entre el área de pruebas y desarrollo, se logró disminuir el porcentaje en 15 errores equivalentes al 19.3% y 0 sugerencias, es decir se logró erradicar ese aspecto, que a pesar de no ser de un alto nivel de criticidad si se tomaron en cuenta las sugerencias hechas por el área de pruebas.

Finalmente, en una tercera iteración, dando seguimiento mediante el procedimiento expuesto en el modelo de la presente investigación, se logró obtener el grado de satisfacción deseado, obteniendo un total de 0 errores y 0 sugerencias, gracias al proceso y plantillas ocupados de la metodología propuesta.

Como resultados finales en base a la investigación y resultados obtenidos en la aplicación del modelo de pruebas a los sistemas mencionados, se obtuvieron los siguientes números relevantes:

Proyecto	Errores	Solucionado	Producto liberado
Composición de relaciones.	44.7 %	100 %	SI
Operaciones con conjuntos difusos.	31 %	100 %	SI

Tabla 32. Resultados concentrados (Elaboración propia).

Con esto se dio por concluido y liberado el producto y terminado el proceso de pruebas, logrando verificar y validar que el software cumple con las características y calidad deseada.

5.2 Análisis de la metodología propuesta

Para poder expresar que la metodología propuesta es una buena opción en comparativa con algunas existentes y mencionadas en el capítulo 1 en el apartado de "Análisis del estado del arte", es importante tomar en cuenta algunos aspectos y cualidades de todos los modelos expuestos, en base a lo mencionado se podrá determinar el grado de satisfacción y herramientas proporcionadas por cada uno de ellos para determinar la mejor opción para la ejecución de pruebas al software.

Este punto es indispensable para poder determinar e indicar si un modelo cumple con las expectativas y objetivos de las pruebas, así como las herramientas y procesos definidos se adaptan a todos los tipos de software y la facilidad de poder ejecutar todo el proceso con el mínimo de conocimientos en el área de pruebas.

Para ello se tomará en cuenta los siguientes puntos a evaluar:

- Cantidad de pasos para generar casos de prueba.
- Proceso definido para generación, ejecución, seguimiento y cierre de pruebas.
- Ejemplo de tipos de datos a utilizar en las pruebas.
- Proceso para seguimiento de bugs.
- Clasificación y nivel de errores.
- Criterios de aceptación para liberar un producto.
- Nomenclatura para identificar un documento y/o componente.
- A partir de requisitos, diseño o casos de uso generar casos de prueba.
- Estimación de tiempos.
- Identificación de participantes y responsabilidades.

- Plan de comunicación entre involucrados.
- Plantillas para llevar a cabo el proceso de pruebas.

A continuación se presenta una tabla con las características mencionadas y que exponen cuales de los modelos existentes logran cumplir con dichos aspectos.

Característica / modelo	SCENT	Heumann	Riebish	AGEDIS	PROPUESTA
Pasos para generar casos de prueba.	19	3	6	6	5
Proceso definido para generar, ejecutar, dar seguimiento y cierre de pruebas.	X	X	X	X	X
Ejemplo de tipos de datos a utilizar en las pruebas.					X
Proceso para seguimiento de bugs.	X	X		X	X
Clasificación y nivel de errores.					X
Criterios de aceptación para liberar un producto	X	X	X	X	X
Nomenclatura para identificar un documento y/o componente					X
A partir de requisitos, diseño o casos de uso, generar casos de prueba					X
Estimación de tiempos.					X
Identificación de participantes y responsabilidades.					X
Plan de comunicación entre involucrados					X
Plantillas para llevar a cabo el proceso					X

Tabla 33. Evaluación de metodologías (Elaboración propia).

Como se puede observar en la tabla 30, se tomaron en cuenta aspectos relevantes en el proceso de desarrollo de pruebas de software, aspectos que son indispensables que un modelo tenga para facilitar al probador el desarrollo de una actividad con las herramientas y métodos bien definidos. La mayoría de las metodologías existentes describen un proceso de pruebas, sin embargo el tener un método para poder guiar la actividad de pruebas, no implica sea el correcto y facilite que el proceso se pueda llevar a cabo de forma adecuada y con las herramientas pertinentes.

La información presentada nos da un panorama general y a su vez específico de los elementos con los que cuenta cada metodología, con ello se pone en manifiesto la importancia de definir, presentar y guiar un buen proceso de pruebas de software que llevara a la reducción de tiempos en el área. Es bien sabido que el costo total de una mala planeación y ejecución de pruebas, repercute en aspectos de pérdida y robo de información, costos elevados de mantenimiento y desfase en la entrega de productos, al grado de generar un costo total de entre 30-40% de todo el proyecto.

Así mismo en tener un proceso bien definido, con herramientas y elementos indispensables, genera que el costo en el mantenimiento se reduzca hasta en un 50%. En base al análisis realizado a los modelos existentes versus el propuesto, se encontraron aspectos relevantes que el resto no tiene y que conllevan a realizar un proceso bien definido, entre estos se encuentran:

- Presentación de plantillas para los diversos tipos de pruebas.
- Presentación de plantilla para plan de pruebas.
- Inclusión de pruebas funcionales, integrales, pruebas de carga, pruebas de rendimiento y seguridad en un solo modelo.
- Tabla de valores como guía.
- Propuesta de herramienta para el registro de errores.
- Propuesta de seguimiento y cierre de incidencias encontradas en las pruebas.

Estos aspectos hacen de la metodología no solo un modelo de pruebas, si no también, un conjunto de pasos que brinda de herramientas y plantillas que servirán en la ejecución de escenarios para el cumplimiento de los objetivos de la empresa, así como de ser una propuesta que puede ser incorporada a cualquier organización y ser utilizada por cualquier persona con o sin conocimientos especializados en la materia. Además de ser un proceso flexible por permitir incorporar elementos que en su momento se requieran o bien sustituir por otros con el mismo fin como es el caso del seguimiento y reporte de pruebas.

Con todo lo mencionado anteriormente, no significa que las metodologías existente sean malas o no cumplan con las expectativas de las empresas, simplemente existen puntos adicionales que podrían considerarse para tener un mejor rendimiento en el proceso de pruebas.

Capítulo 6 Conclusiones y recomendaciones

Una vez que se han revisado las referencias teóricas que involucran las pruebas de software y los elementos implicados en la creación de la metodología propuesta, así como una breve comparativa entre los modelos existentes, en este capítulo se desglosan algunas reflexiones generadas a lo largo de la indagación, tomando como referencia los resultados obtenidos de la aplicación de dicho modelo a dos proyectos de software que fueron presentados en el capítulo 4 y los resultados y análisis puntualizados en el capítulo 5. Así mismo se responde a la pregunta de investigación presentada en este documento y razón por la cual se llevara a cabo la presente averiguación al tema.

Finalmente se presentan las recomendaciones para futuras exploraciones como punto de partida de los resultados y productos generados en dicha investigación, mismos que podrían mejorarse y automatizarse para cumplir aspectos cambiantes en el mercado y área de la ingeniería de software.

6.1 Conclusiones

Se parte de la pregunta de investigación ¿Es posible la implementación de un modelo de pruebas de software como medio de verificación y validación de productos de calidad, que garantice la liberación de productos de software de alta competitividad?, se da una respuesta a través de la confrontación de los resultados obtenidos y analizados en las sección 5.1 y 5.2 de la presente investigación, donde se tomaron dos aspectos fundamentales para llegar al resultado mostrado.

Como primer aspecto, se tomó el grado de facilidad y factibilidad en la creación de las pruebas a dos sistemas de software, así el porcentaje de errores encontrados y

solucionados gracias al modelo propuesto, resultando estos satisfactorios para la liberación y validación de productos de calidad.

Como segundo punto, se tomó la cantidad de elementos y herramientas que proporcionaba cada modelo, con eso se respalda que la propuesta presentada tiene características que llevan al cumplimiento, mejora y facilidad de uso en la generación, ejecución, seguimiento y cierre de errores en el proceso de pruebas.

Así mismo, un aspecto muy importante que se tomó en la comparación de estos modelos, fue no solo tener un proceso definido para cada una de las etapas en las pruebas de software, sino también, guiar y presentar plantillas para cada una de estas, así como una breve descripción del ¿Qué hacer? y ¿Cómo hacer? cada actividad, aspecto que en modelos anteriores no se tomaba en cuenta y que logran solucionarse y abatirse con las herramientas presentadas por la metodología propuesta.

Finalmente cabe recalcar que cada aspecto presentado y expuesto en esta metodología, ayuda a disipar dudas respecto a cada uno de los elementos contenidos en la investigación, además que la factibilidad y flexibilidad que tiene el modelo al ser capaz de trabajar no solo con herramientas especializadas para el seguimiento de fallos como lo es *Bugzilla*, sino también, con simples documentos con los cuales la creación, ejecución, seguimiento y cierre de errores en el proceso de pruebas de software se lleva de forma adecuada y con procesos bien definidos, lo que llevara a cumplir con normas y lineamientos establecidos por instituciones certificadores en el diseño y desarrollo de sistemas de software si así se requiere.

Por tanto el objetivo principal de proponer una metodología para la generación y ejecución de pruebas como medio de verificación y evaluación para software de calidad, se cumple con gran satisfacción, teniendo en cuenta los resultados obtenidos para satisfacer con criterios de modelos de desarrollo de software, como CMMI, Moprosoft, TSP y PSP, implantados en las organizaciones y empresas en general, al ser concluida de forma satisfactoria con resultados relevantes en la aplicación de dos proyectos de software.

Por último a manera de conclusión general, la metodología de pruebas planteada en el presente trabajo de investigación, pudo adaptarse de manera satisfactoria con el proceso que se lleva a cabo en el área de desarrollo de los proyectos evaluados, lo que favoreció en la elaboración, ejecución, seguimiento y cierre de pruebas para obtener resultados satisfactorios y que concluyeron en la liberación de un producto de con características deseadas por parte del área de calidad, cumpliendo con los objetivos planteados por la organización. Por lo que podemos decir que, una buena planeación y metodología en la etapa de pruebas es parte fundamental para que el proceso se lleve a cabo de forma adecuada, lo que contribuirá a encontrar la mayor cantidad de defectos en los productos en el menor tiempo y esfuerzo posible, logrando el cumplimiento de los requisitos y objetivos de los proyectos.

6.2 Trabajos futuros

Es importante mencionar que la presente de investigación pretende ser la base para futuras exploraciones en el área de ingeniería de software con temas selectos como calidad y pruebas a sistemas, gracias a la presentación de procesos actuales, definición de metodologías existentes y herramientas que pueden ser ocupadas para la cumplir con el objetivos de estas dos grandes ámbitos y que día con día se requiere que sean más especializadas, perfeccionistas y detalladas para facilitar su aplicación dentro de las organizaciones y empresas involucradas con tecnologías de la información y comunicación.

A continuación se exponen las futuras líneas de investigación que fueron surgiendo durante la elaboración de la presenta indagación:

1. Realizar un trabajo de análisis que complemente la metodología propuesta, para poder incluir Pruebas de Aceptación, Pruebas de estreés, Pruebas de Concurrencia, Pruebas de Regresión, Pruebas de Desempeño, entre otras.
2. Generar una aplicación para que el modelo propuesto que pueda integrar la automatización de las pruebas mencionadas en la presente investigación.
3. Identificar y probar alguna otra herramienta para el seguimiento de errores que pueda ser adoptada y adaptada por la presente metodología con mayor facilidad y factibilidad.

Con esto damos por concluido y terminada la presente investigación, esperando poder servir como base de futuros descubrimientos y experimentos en el proceso de ingeniería de software que lleven a la generación de más y más productos de alta calidad y competitividad que cumplan con la satisfacción al 100 % de lo que el cliente final espera obtener de un sistema informático.

Bibliografía

Beizer B. (1990). "Software testing techniques (2nd ed.)", ISBN: 0-442-20672-0, Van Nostrand Reinhold Co, 1990.

Boehm, B., (2001) "The Spiral Model as a Tool for Evolutionary Software Acquisition", CrossTalk, mayo 2001, recuperado de: www.stsc.hill.af.mil/crosstalk/2001/05/boehm.html.

Bugzilla. (2015). Recuperado de: Bugzilla https://www.bugzilla.org/about/

"Capability Maturity Model Integration (CMMI)", Version 1.1 (2002). CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1), Continuous Representation CMU/SEI-2002-TR-011 ESC-TR-2002-011, 2002.

Causevic, A., Sundmark, D. and Punnekkat, S. (2010). An Industrial Survey on Contemporary Aspects of Software Testing. 2010 Third International Conference on Software Testing, Verification and Validation, 393 – 401.

Díaz, José, Roberto. (2009) Las metodologías ágiles como garantía de calidad del software REICIS Revista Española de Innovación, Calidad e Ingeniería del Software, vol. 5,núm. 3, octubre, 2009, pp. 40-43 Asociación de Técnicos de Informática Madrid, España

Justiz, D., Gómez, D., y Delgado, M. D. (2013). Proceso de pruebas para productos de software en un laboratorio de calidad. Ingeniería Industrial, *Vol. XXXV, No. 2, Páginas 131-145.*

Fangchun, J., Yunfan, L. (2012). Software testing model selection research based on Yin-Yang testing theory. (2012) International Conference on Computer Science and Information Processing (CSIP), pages 590 – 594.

Garving, David. (1984) “What Does ‘Product Quality’ Really Mean? Sloan Management Review, 25-45.

Glass, Robert. (1998). “Defining Quality Intuitively”. IEEE Software, 103-104,107.

Hui, Z., Huang, S., Hu, B., Yao, Y. (2010). Software security testing based on typical SSD: A case study. 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE) pages V2-312 - V2-316

Hu, H., Jiang, C., Cai, K. (2008). Adaptive Software Testing in the Context of an Improved Controlled Markov Chain Model. 2008 Annual IEEE International Computer Software and Applications Conference, Pages 853 - 858

International Software Testing Qualifications Board (2005), Certified Tester Foundation Level Syllabus, Versión 2005. http://www.istqb.org/fileadmin/media/SyllabusFoundation.pdf

IEEE. “IEEE guide for software verification and validation plans” (1994). IEEE Std 1059-1993, pages i–87, IEEE Computer Society, December 1994.

IEEE. “IEEE standard glossary of software engineering terminology” (1990). IEEE Std 610.12-1990, pages 1–84, IEEE Computer Society, December 1990.

IEEE Standard for Software and System Test Documentation (2008). IEEE Std 829-2008, IEEE Computer Society, July 2008.

Guide to the Software Engineering Body of Knowledge SWEBOK (2004). IEEE Computer Society. http://www.swebok.org, 2004.

JACOBSON, IVAR; G. BOOCH AND J. RUMBEAUGH. (2001). Use Cases: Yesterday, Today, and Tomorrow, The Rational Edge, March, 2001. Disponible en: http://www.therationaledge.com/ content/mar_03/f_useCases_ij.jsp

Jones, C. (2012) Software defect origins and removal methods. Namcook Analystics LLC, page 2.

P. C. Jorgensen. (2002). Software Testing: A Craftsman's Approach. 2nd edition. CRC Press, Inc., Boca Raton, FL, USA.

Kit E. (1995). "Software Testing In The Real World : Improving The Process", ISBN 0201877562, Addison Wesley.

González, L. (2009), Método para generar casos de prueba funcional en el desarrollo de software. Revista Ingenierías Universidad de Medellín. vol. 8, No. 15 especial, pp. 29-36

McCall, J., Richards, P., & Walters, G. (1977). Factors in Software Quality. Volumes I, II and III: RADC Reports.

Myers G. (2004). "The art of software testing, 2nd edition", ISBN 0-471-46912-2, John Wiley & Sons Inc.

G. J. Myers. (1979). Art of Software Testing. 1st edition. John Wiley & Sons, Inc., New York, NY, USA, 1979.

Nakagawa, E. Y., Maldonado, J.C. (2011). Contributions and Perspectives in Architectures of Software Testing Environments. 2011 25th Brazilian Symposium on Software Engineering, Pages 66 – 71.

Jordán, E., Vázquez, R. (2006). Generación de casos de prueba a partir de casos de uso en las pruebas de software. Ingeniería industrial, Vol. XXVII, No 1.

Potter, B., McGraw, G. (2010). Software Security Testing. 2004 IEEE SECURITY & PRIVACY, pages 81 – 85.

Pressman, R. S. (2010) Ingeniería del Software, un enfoque práctico. Séptima edición. Ed. McGraw Hill, 463.

Qian, H., Zheng, C. (2012) A Embedded Software Testing Process Model. 2009 CISE International Conference Computational Intelligence and Software Engineering., pages 1-5.

Sommerville I. (2011) Ingeniería del Software. Novena edición. Ed. Pearson Educación, Pag. 206.

Sommerville. I. (2005). Ingeniería del software. Séptima edición. Pearson Educación.

Triou, E., Abbas, Z., Kothapalle S. (2009). Declarative Testing: A Paradigm for Testing Software Applications. 2009 Sixth International Conference on Information Technology: New Generations, pages 769 - 773.

Wu, X., Sun, J., (2010). The Study on an Intelligent General-Purpose Automated Software Testing Suite. 2010 International Conference on Intelligent Computation Technology and Automation, pages 993 - 996.

Xue-Mei, L., Guochang, G., Yong-Po, L., Ji, W. (2009). Research and Implementation of Knowledge Management Methods in Software Testing Process. 2009 World Congress on Computer Science and Information Engineering, Pages 739 - 743.

Zhang, B., Shen, X. (2011). The Effectiveness of Real-time Embedded Software Testing. 2011 9th International Conference on Reliability, Maintainability and Safety (ICRMS), pages 661-664.

Apéndice

A. Plantilla para plan de pruebas

Nomenclatura para nombrar al archivo Plan de pruebas:

[ACRÓNIMO] - Plan de Pruebas

A continuación se presenta la plantilla a utilizar para el Plan de Pruebas, mismo que será quien regirá toda la etapa de pruebas.

Fecha:

/ / /

Documento Plan de Pruebas para el Sistema

Composición de Relaciones

Versión 2.0

Autor:

Tabla de contenido

SECCIÓN I: CONTROL

1.1.- Revisiones

Realizo		**Fecha**	
Aprobó		Fecha	

1.2.- Versiones

Versión	Descripción	Autor	Fecha

SECCIÓN II: INFORMACIÓN ESENCIAL

2.1.- Introducción

Una de las principales inquietudes en el área de Pruebas es la forma en cómo se realizaran las pruebas a los componentes de cada software, así mismo las herramientas necesarias para llevar acabo dichas tareas. El contenido de este documento es parte integral de una metodología de pruebas para la creación y ejecución de estas como medio de validación de productos de calidad.

El propósito de este documento el cual contiene el Plan de Pruebas, es regir y planificar todas las etapas de las Pruebas Funcionales, Carga, Rendimiento, Integrales y Seguridad. El Plan no solo gobierna todas las etapas si no también establece las técnicas, herramientas y actividades relacionadas con la ejecución y validación de cada una de las etapas, incluyendo responsabilidades de cada una de las actividades, los recursos y los prerrequisitos que deben ser considerados en el esfuerzo de cada una de ellas; esto con la finalidad de llevar un correcto seguimiento de inicio a fin de este proceso y garantizar la operatividad y funcionalidad de la solución desarrollada en base a los requerimientos planteados.

2.2.- Objetivos

2.2.1.- Objetivo general

El presente documento tiene como objetivo diseñar, ejecutar y guiar las pruebas para descubrir la mayor cantidad de errores en el menor tiempo y esfuerzo posible.

2.2.2.- Objetivos específicos

Los principales objetivos de realizar las pruebas son:

- Probar la funcionalidad por módulos del software conforme los escenarios definidos.
- Probar la funcionalidad integral del software conforme los escenarios definidos.
- Probar la funcionalidad de la seguridad del software conforme los escenarios definidos.

- Probar si el software hace lo que no debe, es decir, si provoca efectos indeseados.
- Encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles.
- Mostrar hasta qué punto las funciones del software operan de acuerdo con las especificaciones y requisitos del cliente.
- Establecer el alcance de las pruebas que se realizarán según su tipo y tiempo.
- Determinar los grupos que van a participar en la creación y ejecución de las Pruebas.
- Indicar los Criterios de Aceptación del software en la ejecución de las Pruebas.
- Definir herramientas que se utilizarán en la ejecución de los diferentes tipos de pruebas.
- Especificar la forma y medios para registrar los resultados de la ejecución de pruebas.

2.3.- Alcance

El presente documento lista las distintas pruebas a ser ejecutadas durante el proceso de prueba del sistema que permitirán asegurar el cumplimiento de los objetivos y los requerimientos del software, la forma en la que se ejecutaran, los criterios de aceptación, así como la forma y medios para registrar los resultados de la ejecución de pruebas.

Dentro de los alcances de las pruebas al sistema se establecen los siguientes puntos:

- Pruebas Funcionales
- Pruebas Integrales
- Pruebas de seguridad
- Pruebas de carga
- Pruebas de rendimiento

Así mismo cabe mencionar en base a lo anterior, dicho plan forma parte integral de la metodología basada en tres etapas:

4. Diseño de pruebas.
5. Construcción y ejecución de pruebas que se adapte a la metodología de desarrollo de software que la empresa u organización determine.
6. Seguimiento y resultados.

Dentro de cada una de las tres principales etapas mencionadas se divide en subetapas que apoyaran al cumplimiento de los objetivos de la metodología propuesta. A continuación se describe la clasificación de cada una de ellas para posteriormente ser explicada detalladamente:

4. Diseño de pruebas
    1. Gestión de plantillas para pruebas
    2. Identificación de requerimientos y casos de usos del sistema
    3. Generación de Plan de pruebas
5. Construcción y ejecución de pruebas
    1. Identificación, clasificación y eliminación de casos de prueba redundantes
    2. Creación de casos y escenarios de prueba al sistema

3. Ejecución de casos y escenarios de prueba al sistema

6. Seguimiento y resultados
    1. Reporte y seguimiento de incidencias encontradas en la ejecución
    2. Cierre y resultados finales de incidencias encontradas
    3. Generación de reporte final y análisis

Finalmente es importante señalar que cada una de dichas pruebas serán aplicadas a los módulos correspondientes o al sistema en general, esto teniendo un enfoque de pruebas de caja negra, donde se analizaran la entrada de los datos y las salida de la información para comparan con los objetivos preestablecidos.

2.4.- Fuera del alcance

Quedan fuera del alcance todos aquellos componentes y objetivos que no estén descritos en la especificación de requerimientos del sistema, así como en el presente documento Plan de Pruebas.

2.5.- Modelo de pruebas

La estrategia de pruebas está basada en el proceso especificado en la investigación “Modelo para la generación y ejecución de pruebas como medio de verificación y validación de productos de software de calidad” para todas aquellas empresas enfocadas al desarrollo de software, donde se definen los lineamientos y procedimientos, así como el análisis y justificación de dicho método para asegurar el éxito de los componentes y productos a validar, cuyas actividades macro son:

- Planificación de las pruebas.

- Seleccionar los componentes y productos a probar.
- Realizar los escenarios de prueba con su correspondiente información a implementar.
- Establecer el ambiente para las pruebas.
- Establecer la versión a probar.
- Ejecutar las pruebas de la solución tecnológica.
- Documentar los defectos, hallazgos y no conformidades detectados en la solución tecnológica.
- Analizar la resolución de defectos, hallazgos y no conformidades.

SECCIÓN III: ADMINISTRACIÓN

3.1.- Documentos base

Nombre del documento	Versión

3.2.- Escritos a entregar

Nombre del documento	Fecha de entrega	Entregado

3.3.- Participantes

Integrantes de pruebas:

Nombre / Rol	Área	Actividad

Integrantes de programadores:

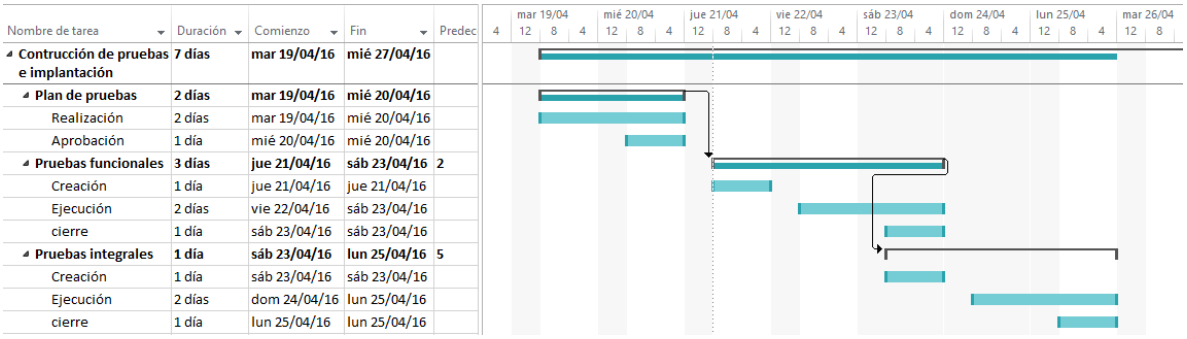
Nombre / Rol	Nombre	Actividad

Los Recursos Materiales para la ejecución de las pruebas:

Cantidad	Recursos	Disponible

3.4.- Cronograma de actividades

A continuación se especifica el calendario de las actividades correspondiente a la creación y ejecución de las pruebas para el sistema.



Nombre de tarea	Duración	Comienzo	Fin	Predec
Contrucción de pruebas e implantación	**7 días**	**mar 19/04/16**	**mié 27/04/16**	
Plan de pruebas	**2 días**	**mar 19/04/16**	**mié 20/04/16**	
Realización	2 días	mar 19/04/16	mié 20/04/16	
Aprobación	1 día	mié 20/04/16	mié 20/04/16	
Pruebas funcionales	**3 días**	**jue 21/04/16**	**sáb 23/04/16**	**2**
Creación	1 día	jue 21/04/16	jue 21/04/16	
Ejecución	2 días	vie 22/04/16	sáb 23/04/16	
cierre	1 día	sáb 23/04/16	sáb 23/04/16	
Pruebas integrales	**1 día**	**sáb 23/04/16**	**lun 25/04/16**	**5**
Creación	1 día	sáb 23/04/16	sáb 23/04/16	
Ejecución	2 días	dom 24/04/16	lun 25/04/16	
cierre	1 día	lun 25/04/16	lun 25/04/16	

SECCIÓN IV: DATOS TÉCNICOS

4.1.- Elementos a probar

Resultados

Componentes a probar	Tipo de prueba a ejecutar	Satisfactorio

4.2.- Control de riesgos

Id	Fecha de descubrimiento	Severidad	Solución	Fecha de solución

4.3.- Niveles en los defectos

A continuación se describe la severidad de los defectos para su clasificación, esto con la finalidad de determinar el grado al que pertenece y su correspondiente acción de corrección.

Dentro de los defectos se encuentras los siguientes:

Critica:

- Impide visualizar los resultados esperados de la prueba por alguna falla en el sistema o en la configuración.
- Existen enlaces rotos, inválidos o no implementados.
- El sistema no responde y no hay alternativas para ejecutar las pruebas.

- El sistema se cierra inesperadamente.

Alta:

- No cumple con los requerimientos.
- Inconsistencia de datos.
- Secciones de un requerimiento faltantes.
- Mal funcionamiento de algún método o complemento.

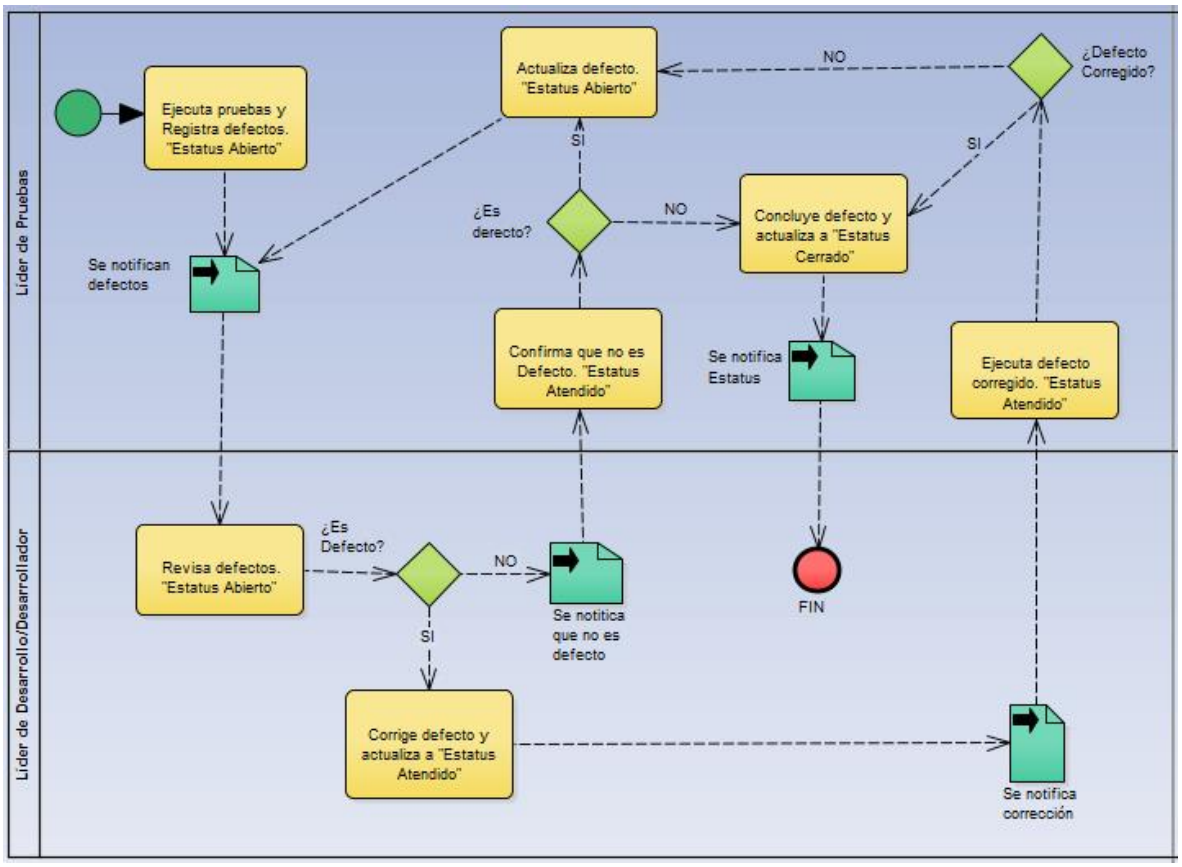
Media:

- Campos o pantallas faltantes.
- Errores que permitan continuar con el proceso.
- No muestra información completa y correctamente.
- El sistema no envía mensajes al usuario.

Baja:

- Errores ortográficos o de diseño.
- El sistema no indica que son campos obligatorios con (*) o notación resaltada.

4.4.- Ciclo de vida de los defectos



4.5.- Ejecución de Pruebas

A continuación se señalan las condiciones mínimas que se deben presentar para iniciar la ejecución de las pruebas:

- Se poseen los escenarios de pruebas aprobados.
- El entorno de pruebas es el adecuado para el tipo de pruebas a iniciar.
- Los materiales se encuentran disponibles.
- Se recibió la Versión del Software para pruebas con su correspondiente ambiente.
- Todos los recursos humanos y técnicos necesarios se encuentran disponibles.

4.6.- Suspensión de Pruebas

A continuación se describen los criterios por los cuales se podrá suspender la ejecución de las pruebas:

- Indisponibilidad del ambiente para verificación y validación.
- No contar los escenarios de pruebas.
- No contar con los requerimientos necesarios para la ejecución del servicio (Software, Hardware, documentación etc.).
- No contar con la versión final del sistema.
- Indisponibilidad de herramientas para la ejecución del servicio.
- No contar con todos los recursos humanos y técnicos necesarios.

4.7.- Reanudación de pruebas

A continuación se indican los criterios por los cuales se podrá reanudar la ejecución de las pruebas:

- Disponibilidad de ambiente para verificación y validación.
- Contar con los requerimientos necesarios para la ejecución del servicio (Software, Hardware, documentación etc.).
- Contar con la versión final del sistema.
- Disponibilidad de herramientas para la ejecución del servicio.
- Todos los recursos humanos y técnicos necesarios se encuentran disponibles.

4.8- Validación y aceptación

A continuación se indican los criterios por los cuales se podrá dar por satisfactoria la validación de Calidad:

- Que el número de defectos sea menor al 1% por módulo y que su severidad no sea crítica, alta o media, y que estos no afecten el desempeño del sistema para continuar con el siguiente tipo de prueba.
- Que el número final de defectos en suma de todos sus componentes no sea mayor al 2% y que su severidad no sea crítica, alta o media, y que estos no afecten el desempeño del sistema.

Solo en estos casos se podrá dar por validada una prueba y el sistema en general para la liberación por parte del área de TESTEO.

4.9.-Revisión de documentos

Para la revisión de todos los documentos de prueba se tendrá un lugar destinado el cual será determinado por el líder del proyecto donde contendrá todos los entregables para su validación.

4.10.- Comunicación

Actor 1 / Rol	Actor 2 / Rol	Medio de comunicación	Evidencia

4.11.- Reporte de hechos

Id	Incidencia	Fecha	Responsable / Rol

4.12.- Definiciones

Pruebas unitarias: Las pruebas unitarias verifican si cada uno de los componentes de forma independiente funciona correctamente a través de su interfaz, cubren la funcionalidad establecida, y se ajustan a los requisitos especificados en las verificaciones correspondientes.

Pruebas funcionales: Las pruebas de integración verifican si los componentes o subsistemas en su conjunto interactúan correctamente a través de su interfaz, cubren la funcionalidad establecida, y se ajustan a los requisitos especificados en las verificaciones correspondientes.

Pruebas integrales: Se prueban las funcionalidades del sistema con las relaciones a los módulos definidos.

Pruebas de rendimientos: Las Pruebas de Rendimiento se ejecutan tanto para determinar cómo responde un sistema ante una cierta carga, como para validar otros atributos relacionados con la calidad, como pueden ser la escalabilidad, la fiabilidad o el uso de recursos entre otros.

Pruebas de seguridad: Las Pruebas de Seguridad pretenden medir la Confidencialidad, Integridad y Disponibilidad de los datos tratando de identificar amenazas y riesgos desde el uso o interface de usuario final.

Pruebas de caja negra: Se prueban las funcionalidades de cada módulo con las respectivas entradas de datos y salidas de información.

*El plan de prueba: D*escribe todos los métodos que se utilizarán para verificar que el software satisface la especificación del producto y las necesidades del cliente. Incluye los objetivos de calidad, necesidades de recursos, cronograma, asignaciones, métodos, etc.

Casos de prueba: Lista los ítems específicos que serán probados y describe los pasos detallados que serán seguidos para verificar el software.

Reporte de pruebas: Describen los problemas encontrados al ejecutar los casos de prueba.

Herramientas de pruebas y automatización: Documentación de las herramientas empleadas en el proceso de pruebas.

Métricas, estadísticas y resúmenes: Indican como ha sido el progreso del proceso de prueba.

Estrategia: Serie de acciones/actividades a realizar para establecer un marco de trabajo.

Verificación: Actividad para revisar si un entregable o producto es correcto porque cumple con la especificación de su diseño.

Producto: Documento que es requerido en cada una de las fases y/o documentación con la que debe cumplir un rol determinado.

Incidencia: Documento que es requerido en cada una de las fases y/o documentación con la que debe cumplir un rol determinado.

SECCIÓN V: AUTÓGRAFOS

5.1.- Firmas

Por parte del área de Pruebas:

Cargo
[nombre]

Por parte del área de Desarrollo:

Cargo
[nombre]

B. Matriz de datos para pruebas

Nomenclatura para nombrar al archivo con los escenarios de pruebas:

(ACRÓNIMO)-(MÓDULO)-(TIPO DE PRUEBA) - (NOMBRE DEL CASO DE PRUEBA)

A continuación se presenta la plantilla a utilizar para general las pruebas, mismo que será utilizada para Pruebas Funcionales, Pruebas Integrales, Pruebas de carga, Pruebas de rendimiento y Pruebas de Seguridad según sea el tipo de proyecto.

Registro de revisiones				
versión	Fecha	Autor	Revisor	Observaciones

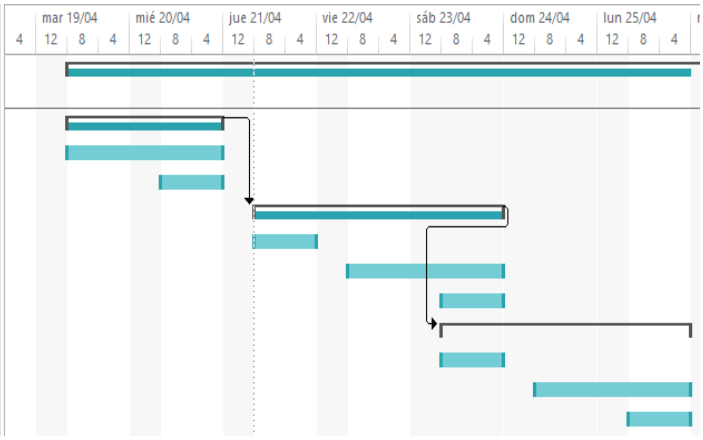
MATRIZ DE DATOS – ESCENARIO DE PRUEBA [FUNCIONAL, INTEGRAL, SEGURIDAD, CARGA, RENDIMIENTO]
[ACRÓNIMO DEL SISTEMA] - M/[No. De Módulo] – CU/[No. De Caso de Uso] - [Nombre del caso de prueba]

INFORMACIÓN DETALLADA	
Identificador del caso de prueba	O] - M/[No. De Módulo] – CU/[No. De Caso de Uso] - ECP/[No De caso de Prueba] - [Nombre del caso de prueba] - [Nombre del escenario
Ruta	Ruta correspondiente para la ejecución del escenario de prueba
Precondiciones	Precondiciones a cumplir antes de iniciar la ejecución de los escenarios de prueba
Versión del sistema a Probar	Versión del sistema a probar

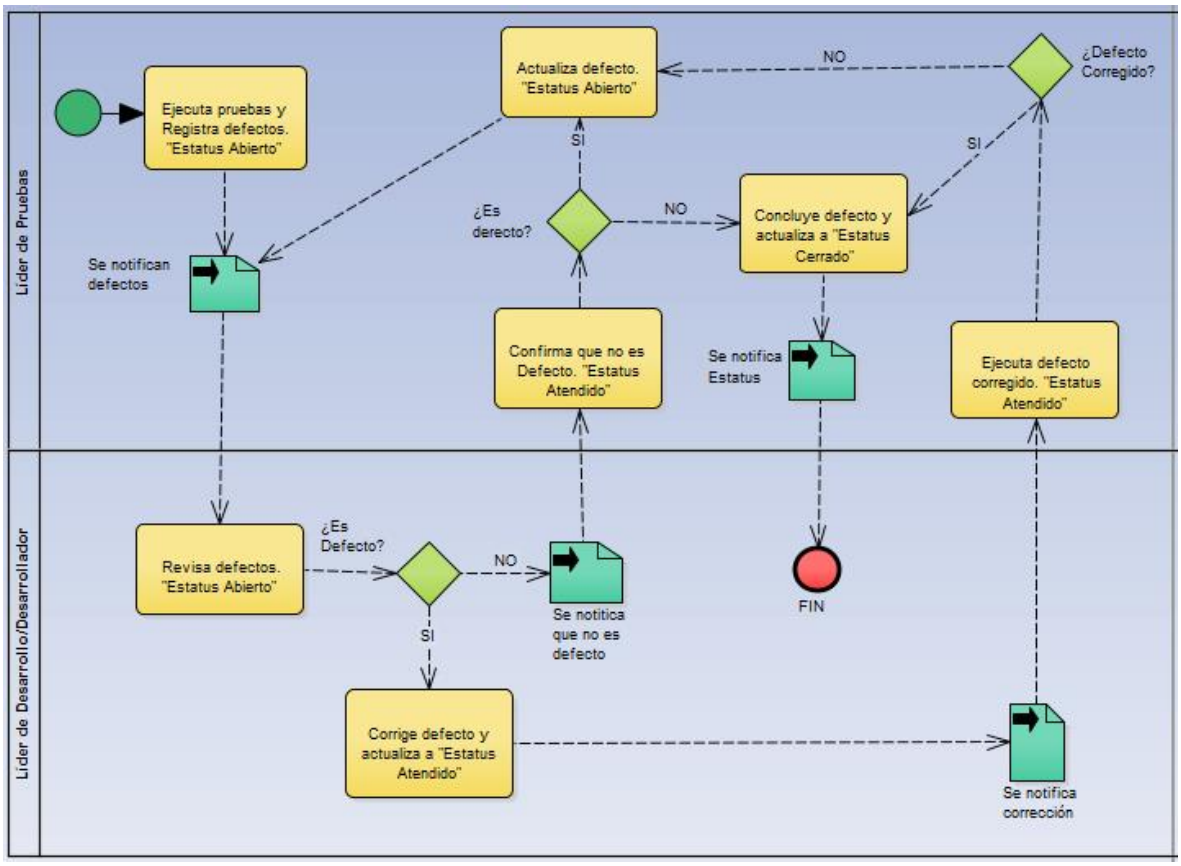
ID	Acción	Datos de entrada	Tipo de Prueba	Resultado Esperado	Resultado Final	Corregido
[ACRÓNIMO] - M/[No. De Módulo] - CU/[No. De caso de uso] -ECP/[No. De Escenario de caso de prueba] - C/[No. De caso]	[Descripción de lo que se realizara en el escenario]	[Información que se utilizara en campos donde se requiera información]	[Optimista / No optimista]	[Descripción del resultado esperado al ejecutar el escenario de prueba]	[Correcto / Incorrecto	[SI, NO, NO APLICA] En caso de no aplicar, es importante una breve descripción

C. Cronograma de actividades

Nombre de tarea	Duración	Comienzo	Fin	Predec
Contrucción de pruebas e implantación	**7 días**	**mar 19/04/16**	**mié 27/04/16**	
Plan de pruebas	**2 días**	**mar 19/04/16**	**mié 20/04/16**	
Realización	2 días	mar 19/04/16	mié 20/04/16	
Aprobación	1 día	mié 20/04/16	mié 20/04/16	
Pruebas funcionales	**3 días**	**jue 21/04/16**	**sáb 23/04/16**	**2**
Creación	1 día	jue 21/04/16	jue 21/04/16	
Ejecución	2 días	vie 22/04/16	sáb 23/04/16	
cierre	1 día	sáb 23/04/16	sáb 23/04/16	
Pruebas integrales	**1 día**	**sáb 23/04/16**	**lun 25/04/16**	**5**
Creación	1 día	sáb 23/04/16	sáb 23/04/16	
Ejecución	2 días	dom 24/04/16	lun 25/04/16	
cierre	1 día	lun 25/04/16	lun 25/04/16	



D. Diagrama de ciclo de vida de errores



Anexos

softura solutions

Softura Solutions S. de R.L.
RFC SSO060801467

Dependencia: Softura Solutions S. de R.L.
Asunto: Carta de Satisfacción

APIZACO, TLAXCALA A 12 DE JUNIO DEL 2015.

MTRO. FELIPE PASCUAL ROSARIO AGUIRRE
DIRECTOR
INSTITUTO TECNOLÓGICO DE APIZACO
P R E S E N T E.

A través de este medio informo a usted que el Lic. **Said Pérez Flores** alumno de la maestría en **Sistemas Computacionales**, con No. de control **M09370370** terminó de forma satisfactoria el Proyecto que llevaba por nombre **"Modelo para la generación y ejecución de pruebas como medio de verificación y validación de productos de software de calidad"**, así mismo hacemos del conocimiento de la Institución que dicho tema por necesidades de la empresa fue cambiado por el que originalmente se tenía como **"Modelo de evaluación y pruebas para garantizar la calidad de software"**. Durante su Estancia Técnica en esta Empresa desarrollo sus actividades en el área de **PRUEBAS,** cuyo responsable corresponde a cargo del **M.C. Germán Escobar Alonso** durante el periodo comprendido de **NOVIEMBRE DEL 2014 A JUNIO DEL 2015**, con un horario de 9:00 a.m. a 12:30 p.m. y de 14:30 – 18:00 pm de lunes a viernes.

Sin otro particular por el momento reciba un cordial saludo.

ATENTAMENTE

softura solutions
R.F.C. SSO 060801 467

Ing. Gerardo Garcia Calva
REPRESENTANTE LEGAL

CALLE DE LOS PINOS No. 12 CASA 1 COL ATEMPAN, TLAXCALA TLAX.
Teléfono (246) 1 44 86 48; www.softura.com.mx

Modelo para la generación de pruebas funcionales, integrales, seguridad y navegación como medio de verificación de productos de software

Lic. Said Pérez Flores[1], M. en C. José Juan Hernández Mora[2],
M. en C. María Guadalupe Medina Barrera[3] y M. en C. María Janai Sánchez Hernández[4]

***Resumen*— En la presente investigación se expone una metodología para apoyo en la generación de pruebas funcionales, integrales, navegación y seguridad para el desarrollo de sistemas de software, así como parámetros y procedimiento a utilizar para el desarrollo de casos de prueba. La creación de dicho modelo para la generación de pruebas como medio de verificación de productos de software le servirá a todas aquellas empresas, grupos de trabajo, organizaciones e individuos inmersos en el ámbito de la ingeniería de software, que tienen la necesidad de producir sistemas o aplicaciones con la finalidad de satisfacer la necesidades cambiantes del cliente y el mercado en general.**

***Palabras clave*— Metodología, Pruebas Integrales, Pruebas funcionales, Pruebas de navegación y Pruebas de Seguridad.**

Introducción

Hoy en día el software está relacionado con la mayoría de los ámbitos, por ejemplo: medicina, física, química, entretenimiento, procesos industriales, telecomunicaciones, etc., permitiendo cumplir tareas comunes de forma sencilla y fácil, además de cubrir grandes necesidades a través de sistemas y aplicaciones. Por lo anterior se exige que la calidad y la confianza de los sistemas automatizados deba ser la adecuada a la hora de generar un producto final. Día a día compañías, dependencias y personas optan por expandir sus productos o servicios a través internet, llegando satisfactoriamente una mayor cantidad de individuos de manera más simple permitiéndoles obtener una mayor ganancia.

Se ha sabido de muchos sucesos desfavorables que han hecho que la desconfianza en el uso de productos, sistemas o aplicaciones de software sea cada vez mayor, esto debido a que las empresas al desarrollar nuevos bienes y servicios relacionados con las tecnologías de la información en las diferentes áreas sean de baja calidad o contengan errores, esto ha provocado que la demanda y el incremento en la calidad de los productos en una empresa desarrolladora de software sea fundamental para poder competir, adaptarse y sobrevivir en el mercado para satisfacer las necesidades del cliente.

Según Jacobson (2003) menciona que en la actualidad los errores y defectos que surgen en el software forman parte de una estadística muy importante que afecta a las empresas y usuarios finales, se ha determinado que un software en el cual no se realizan pruebas, entre el 60% y el 80% de todo el esfuerzo se ocupa en mantenimiento dejando el otro 20% en desarrollo, mientras que un software al cual se le aplica un 5% de esfuerzo en pruebas logra disminuir el porcentaje de mantenimiento hasta en un 50%, es decir se logra eliminar un 30% en el costo de manutención.

Es por todo esto que es necesario tener no solo un modelo o metodología bien definida para pruebas, sino también los procedimientos y herramientas necesarias para poder detectar el mayor porcentaje de errores a la hora de ejecutar las pruebas.

Por esto se propone una alternativa para contribuir al desarrollo y ejecución de pruebas para el software basándose en los requerimientos del cliente y los casos de uso para poder general las pruebas y poder lograr obtener el mayor porcentaje de errores posibles al testear y detectar los bugs existentes en los sistemas y aplicaciones.

[1] El Lic. Said Pérez Flores es Alumno de la División de Estudios de Posgrado en el Tecnológico Nacional de México, Instituto Tecnológico de Apizaco, Tlaxcala, MEX saidperez49@gmail.com said_tmd@hotmail.com (**autor corresponsal**)
[2] EL M. en C. José Juan Hernández Mora es Docente del Tecnológico Nacional de México, Instituto Tecnológico de Apizaco, Tlaxcala, MEX jjhmora@itamail.itapizaco.edu.mx
[3] La M. en C. María Guadalupe Medina Barrera es Docente del Tecnológico Nacional de México, Instituto Tecnológico de Apizaco, Tlaxcala, MEX lupita_medina@hotmail.com
[4] El M. en C. María Janai Sánchez Hernández es Docente del Tecnológico Nacional de México, Instituto Tecnológico de Apizaco, Tlaxcala, MEX shmj90@gmail.com

Descripción del Método

A continuación se enlistan y describen algunos conceptos básicos y necesarios que se utilizaran posteriormente y que ayudaran a la comprensión del modelo planteado, así mismo se representarán algunos modelos y métodos existentes para realizar pruebas a sistemas y finalmente pasar a la propuesta que ocupa esta divulgación.

Conceptos básicos

- Caso de prueba: Es un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y postcondiciones de ejecución, desarrollados con un objetivo particular o condición de prueba, tal como ejercitar un camino de un programa particular o para verificar que se cumple un requerimiento especifico (IEEE2, 1990).
- Resultado esperado: Es el comportamiento predicho por la especificación u otra fuente, del componente o sistema a ser probado bajo condiciones especificadas (Istqb, 2005).
- Ciclo de prueba: Es la ejecución del proceso del testing contra una versión identificada del producto a probar (Istqb, 2005).
- Incidente: cuando una falla sucede, puede o no ser evidente para el usuario (cliente o probador). Un incidente es el síntoma, asociado con la falla, que alerta al usuario de la ocurrencia de una falla (Jor, 2002).
- Pruebas unitarias: Las pruebas unitarias, también llamadas pruebas de componentes, se encargan de probar, individualmente, subprogramas, subrutinas o procedimientos en un programa (Mye, 1979).
- Pruebas funcionales: En general, los componentes de software son componentes compuestos constituidos por varios objetos en interacción. Por consiguiente, la prueba de componentes compuestos tiene que enfocarse en mostrar que la interfaz de componente se comporta según su especificación (Sommerville, 2011).
- Pruebas integrales: Las pruebas de integración son una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar los componentes probados de manera individual y construir una estructura de programa que se haya dictado por diseño (Pressman, 2010).
- Pruebas de seguridad: La pruebas de seguridad intenta verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier penetración impropia (Pressman, 2010).
- Datos de entrada: es el conjunto de todos los posibles datos de entrada que puede recibir un programa. Esto incluye las variables globales, los parámetros recibidos por una función o las entradas que puedan ser introducidas externamente (IEEE2, 1990).
- Requerimientos: Los requerimientos del software expresan las necesidades y las restricciones puestas en un producto de software que contribuyen a la solución de un cierto problema del mundo real. Una característica esencial de todos los requerimientos del software es que sean comprobables. Una afirmación es verificable si se puede diseñar un experimento que demuestre o refute la verdad de la sentencia (Beizer, 1990).

El objetivo principal de las pruebas de caja negra es encontrar ítems de las siguientes categorías (Pressman, 2010):

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en acceso a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Modelos existentes

Para llegar al desarrollo del modelo propuesto se realizó un análisis de las formas actuales para pruebas a sistemas de software y así determinar un correcto procedimiento que cumpla con las herramientas y procesos para producir un buen producto. A continuación se describe la revisión de la literatura.

Fangchun (2012) propone un modelo que divide a las pruebas en caja negra y caja blanca, en base a esto plantea un modelo con la combinación de estas dos ramas para probar los componentes. El propósito de las pruebas de software es llevar acabo las operaciones en condiciones preestablecidas, para conocer los errores y evaluar al mismo tiempo la calidad del software en todas las etapas del proceso de desarrollo del software, cubriendo el ciclo completo para obtener un producto final que cumpla con los requerimientos del sistema.

Un segundo autor González (2009) presenta un método estrictamente limitado a pruebas funcionales sobre sistemas de software. Para poder generar los casos de prueba funcional utilizan como datos base los casos de uso para posteriormente mediante plantillas generadas y diagramas, crear escenarios de prueba. Una vez realizado esto implementa un *CheckList* para comprobar que todos los componentes fueron probados. Este proceso consta de seis pasos, el primero es la identificación de los escenarios que serán probados, posteriormente validar que no existen escenarios repetidos, como tercer paso utilizar las plantillas previamente generadas para poder integrar la información, el seguida se ejecuta el proceso y finalmente verifica si existe algún otro escenario nuevo volviendo a pasos anteriores. Para la ejecución de esta actividad presenta una plantilla bien determinada para su apoyo.

Así mismo Justiz (2014) presenta una propuesta basada en un proceso de pruebas de software para un laboratorio de calidad dentro de un ambiente universitario que consisten en cuatro niveles, pruebas iniciales correspondientes a la planeación y pruebas de código, la segunda son las pruebas del sistema, donde se enfoca la validación de pruebas de rendimiento, de seguridad y funcionales, en la tercer etapa están las pruebas de aceptación, que involucra la verificación de pruebas de rendimiento y seguridad, intervención con el cliente y las pruebas alfa, y finalmente están las pruebas de explotación que involucran ya la implantación en un entorno del cliente con la versión del software Beta. Así mismo propone una seria de roles y responsabilidades para el proceso, entre los cuales se encuentra en Líder de laboratorio de pruebas, el Especialista de pruebas, el Representante del Proyecto y el Probador o Tester. Para poder cumplir con el proceso completo, define un flujo de trabajo para el proceso de pruebas y sus respectivos actores con sus responsabilidades que van desde la solicitud hasta su ejecución y conclusión.

Por otro lado Jordan (2006) al igual que Fangchun propone los tipos de pruebas de caja negra y caja blanca, a diferencia que el primero se basa estrictamente en los casos de uso para poder realizar el proceso de pruebas. El autor menciona que los casos de uso determinar aspectos como, el cliente que obtendrá como producto final, al programador que y como tendrá que desarrollar cada módulo para que funcione según lo especificado, al documentador que partes son indispensables redactar y al tester que tiene que probar. Los casos de uso tienen una estructura definida por flujos principales los cuales indican el procedimiento común y flujos alternos los cuales describen alternativas que podrían darse si el común no se ejecutara, estos a su vez se convierten en escenarios de pruebas a ejecutar para validar si su funcionamiento cumple con lo descrito.

Las anteriores propuestas contienen elementos importantes para le generación de pruebas a sistemas de software. Sin embargo existen factores adicionales que bien podrían sustituirse, como por ejemplo, tienen pasos excesivos y procedimientos complejos que impiden un correcto flujo para cumplir con el objetivo.

Modelo propuesto

En esta sección se presenta y describe el modelo propuesto para la generación de pruebas funcionales, integrales, seguridad y navegación, a partir del análisis realizado a diversas metodologías existentes y la experiencia que se ha tenido en el ámbito empresarial como tester, calidad y auditor.

Dicha propuesta podrá ser utilizada para modelos de desarrollo de software como en espiral, cascada, evolutivos, incremental, modelo V, Scrum, RUP, entre otros.

En la figura 1 se presentan las 3 etapas de las que está compuesta dicha metodología:

Figura 1. Etapas del modelo

Dichas fases estas diseñadas para cumplir con normas como la IEEE, CMMI, o MoproSoft en donde se debe tener un proceso definido y completo que abarque la planificación, ejecución, seguimiento y los resultados para el cumplimiento de las actividades dentro del proceso de pruebas.

En la figura 2 se muestran de forma general el proceso de pruebas para poder llevar acabo la tarea de forma adecuada, así como los artefactos base para que se pueda realizar dicha tarea.

Figura 2. Proceso general de pruebas

El modelo propuesto está compuesto de diferentes fases que ayudaran a un correcto proceso de análisis, ejecución, seguimiento y presentación de resultados en el proceso de pruebas, a continuación mencionan dichos pasos:

1. Diseño de pruebas
    a. Gestión de plantillas para pruebas.
    b. Identificación de requerimientos y casos de usos del sistema.
    c. Generación de plan de pruebas.
2. Construcción y ejecución de pruebas
    a. Identificación, clasificación y eliminación de casos de prueba redundantes.
    b. Creación de casos y escenarios de prueba al sistema.
    c. Ejecución de casos y escenarios de prueba al sistema.
3. Seguimiento y resultados
    a. Reporte y seguimiento de incidencias encontradas en la ejecución.
    b. Cierre y resultados finales de incidencias encontradas.
    c. Generación de reporte final y análisis de resultados.

Cada una de estas etapas es de suma relevancia para poder obtener resultados favorables, a continuación se describe brevemente cada una de las fases.

1.- En la etapa de diseño se gestionan plantillas que serán usadas en la siguiente etapa como:

- Documentos plan de pruebas
- Matriz de datos de pruebas unitarias
- Matriz de datos de pruebas funcionales
- Matriz de datos de pruebas integrales
- Matriz de datos de pruebas de seguridad
- Matriz de datos de pruebas de navegación
- Reporte final

En dicha etapa se presenta el plan de pruebas el cual gobernará todo el proceso posterior, así como la identificación de requerimientos para establecer los artefactos que se van a probar.

2.- En la segunda etapa una vez gestionado los documentos, se generan y validan los casos de prueba que se van a ejecutar en la misma fase, teniendo cuidado de no repetir o caer en redundancia al crear cada escenario de prueba, así mismo se recomienda configurar la herramienta bugzilla para un correcto seguimiento en las pruebas.

En el proceso de creación de los escenarios de pruebas es importante tomar en cuenta algunos datos de entrada para diseñar cada prueba, en la tabla 1 se describen los mismos:

Tipo	Ejemplo
Longitud de campos	Máximas y Mínimas, así como campos vacíos.
Tipos de datos	Caracteres especiales, signos de operación, números enteros, números decimales, números negativos, letras, etc.
Campos de texto	Probar con líneas de código del algún lenguaje de programación como saltos de línea, comentarios, asignación de variables, etc.
Campos obligatorios	Dejar campos vacíos.
Campos opcionales	Probar con información y con campos vacíos.
Campos para operaciones	Probar con valores positivos, negativos, decimales o valores grandes.
Guardar imágenes	Verificar tipos de formato que soporta, tamaño y pixeles.
Guardar documentos	Verificar las diversas extensiones que soporta como, .doc, .xsls, .pdf dot, .txt, .rtf, etc.
Vínculos	En caso de tener vínculos validar que apuntan a la dirección correcta.
Visualizar de imágenes	Verificar que las imágenes se muestran de forma clara y en el tamaño definido.
Visualizar de documentos	Verificar que los documentos se visualicen de forma correcta y con la información necesaria.
Descarga de archivos	Verificar que cada archivo sea legible y en el formato que indica será descargado.
Impresión de sitio	Algunos de los portales tienen la opción de impresión, en la cual se debe validar legibilidad y orden.

Tabla 1. Datos de entrada

3.- Finalmente en la última etapa una vez ejecutadas las pruebas, es muy importante darle el seguimiento correspondiente a cada defecto encontrado para su corrección y verificación. Para ello se utilizará una herramienta especializada para dicha tarea llamada Bugzilla, como se mencionó en la etapa pasada, que permite generar reportes de los estados en los que se encuentra un error, así como automáticamente mandar correo tanto al desarrollador como al tester para su notificación, corrección y cierre.

Comentarios Finales

Como consecuencia de la presente investigación, a continuación se presentan las observaciones finales, las conclusiones, así como las recomendaciones o trabajos futuros que se recomiendan realizar como base de este artículo.

Resumen de resultados

En esta sección se presenta un análisis comparativo de los modelos y métodos existentes con la propuesta en cuestión para determinar las mejores características, herramientas y medios para lograr cumplir con las pruebas de software de manera correcta y adecuada. Esto dio como resultado una mejora en los siguientes puntos:

- Presentación de plantillas para las diversos tipos de pruebas.
- Presentación de plantilla para Plan de Pruebas.
- Inclusión de pruebas funcionales, integrales, de navegación y seguridad en un solo modelo.
- Propuesta de herramienta registro de errores.
- Propuesta de seguimiento y cierre de incidencias encontradas en las pruebas.

Conclusiones

Una buena planeación y metodología en las pruebas es la parte fundamental para que este proceso se lleve a cabo de forma adecuada, lo que contribuirá a encontrar la mayor cantidad de defectos en los productos, así como de validar los requisitos que se establecieron al inicio antes de ser liberados al usuario final. Se debe contar con un método que facilite la generación de diversos tipos de casos de prueba, evitando así la redundancia, pérdida de tiempo y liberación de productos con errores.

Este artículo presento un modelo sencillo, con las plantillas necesarias para su construcción y herramientas fundamentales para su ejecución que facilitaran a todas aquellas empresas, organizaciones o individuos a liberar productos de calidad una vez validada la etapa de pruebas.

En publicaciones posteriores se presentaran casos de estudio desarrollados usando el modelo propuesto, con la finalidad de comparar los resultados obtenidos en la detección de errores en productos de software de diversos fines.

Recomendaciones

Como trabajo futuro recomendado a los investigadores interesados en continuar con nuestra labor, se plantea la construcción de una herramienta informática que automatice y facilite el modelo propuesto sin dejar de lado la calidad y características con las que cuenta la presente investigación.

Referencias

Beizer B. "Software testing techniques (2nd ed.)", *Van Nostrand Reinhold* Co, ISBN:0-442-20672-0, 1990.

Fangchun, J., Yunfan, L. Software testing model selection research based on Yin-Yang testing theory. *International Conference on Computer Science and Information Processing (CSIP)*, pages 590 – 594, 2012.

González, L. Método para generar casos de prueba funcional en el desarrollo de software. *Revista Ingenierías Universidad de Medellín*. vol. 8, No. 15 especial, pp. 29-36, 2009.

IEEE. "IEEE standard glossary of software engineering terminology". *IEEE Computer Society,* IEEE Std 610.12-1990, pages 1–84, December 1990.

I. Sommerville. Ingeniería del Software. Novena edición. Ed. Pearson Educación, Pag. 206, 2001.

Istqb, International Software Testing Qualifications Board, *Certified Tester Foundation Level Syllabus*, Versión 2005. http://www.istqb.org/fileadmin/media/SyllabusFoundation.pdf.

Jacobson, I., G. Booch and J. Rumbeaugh. (2003). Use Cases: Yesterday, Today, and Tomorrow, The Rational Edge. Consultada por internet el 6 de enero del 2015. Dirección de internet: Http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf.

Justiz, D., Gómez, D., y Delgado, M. D. Proceso de pruebas para productos de software en un laboratorio de calidad. *Ingeniería Industrial*, Vol. XXXV, No. 2, Páginas 131-145, 2013.

Jordán, E., Vázquez, R. Generación de casos de prueba a partir de casos de uso en las pruebas de software. *Ingeniería industrial*, Vol XXVII, No 1, 2006.

Myers G. "The art of software testing. 2nd edition", ISBN 0-471-46912-2, John Wiley & Sons Inc., 2004.

Pressman, Roger S. Ingeniería del Software, un enfoque práctico. Sexta edición. Ed. MacGraw Hill, 463, 2010.

P. C. Jorgensen. Software Testing: A Craftsman's Approach. 2nd edition, CRC Press, Inc., Boca Raton, FL, USA, 2002.