Universidad de las Ciencias Informáticas

Facultad 3

Título: Estrategia para la aplicación de Pruebas de Caja Blanca y Caja Negra al proyecto Registros y Notarías.

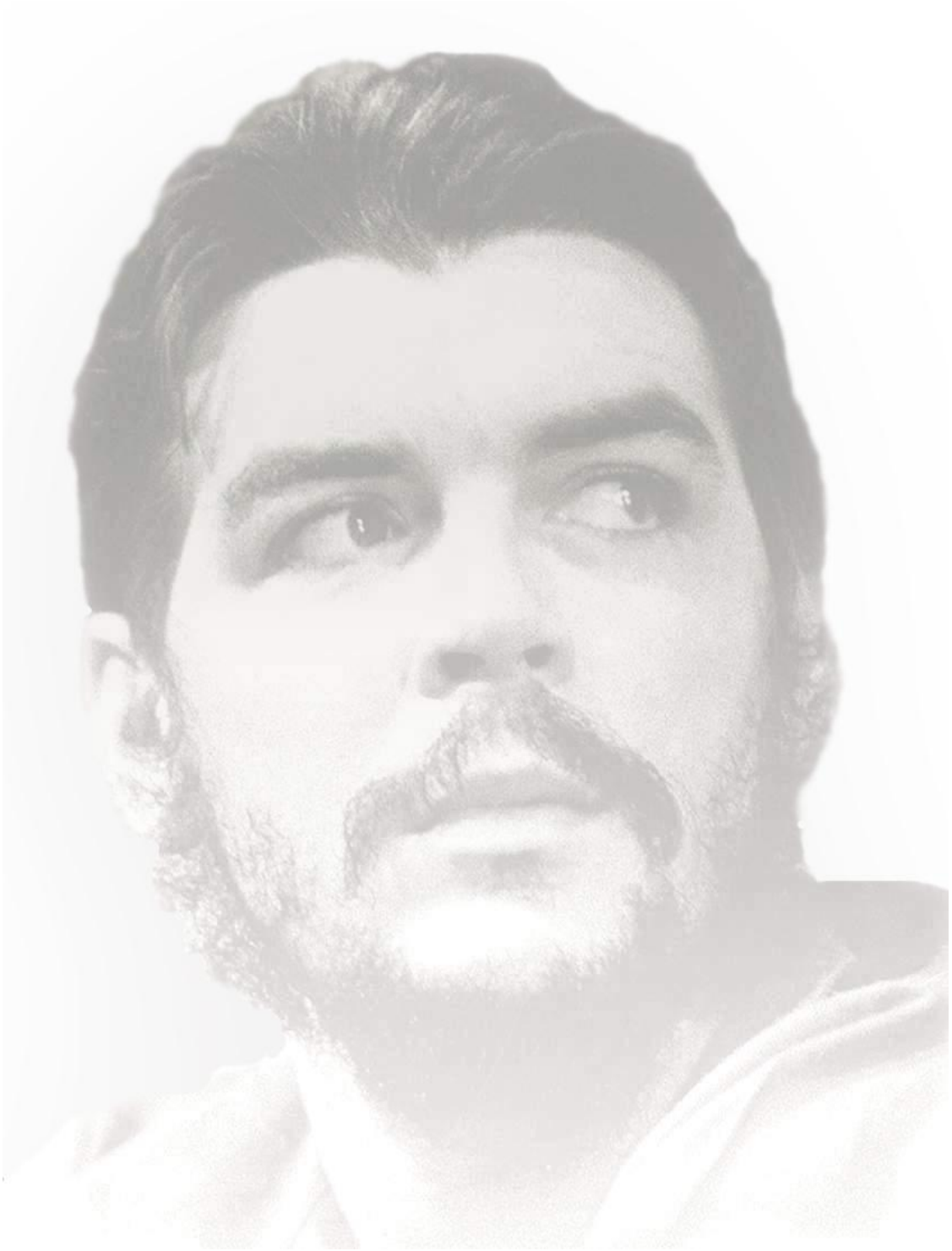
Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor(es): Susana Gonzalez Espinosa.
Dania Durán Cutiño.

Tutor(es): Ing. Heney Díaz Pérez.
Msc. Ing. Maikel Yelandi Leyva Vázquez.

Junio 2008



"Aquí esta una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana."

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Susana Gonzalez Espinosa

_____
Firma del Autor

Dania Durán Cutiño

_____
Firma del Autor

Msc. Ing. Maikel Yelandi Leyva Vázquez

_____
Firma del Tutor

Ing. Heney Díaz Pérez

_____
Firma del Tutor

DATOS DE CONTACTO

Ing. Heney Díaz Pérez.
Especialidad de graduación: Ingeniero en Ciencias Informáticas.
Categoría docente: Adiestrado.
hdperez@uci.cu

Msc. Ing. Maikel Yelandi Leyva Vázquez.
Especialidad de graduación: Ingeniero Informático
Categoría docente: Instructor
Categoría Científica: Máster
Años de experiencia en el tema: 2
Años de graduado: 2
mleyvaz@uci.cu

Agradecimientos

A mi mamá porque sin ella no sería lo que soy, por guiarme siempre, por enseñarme a dar lo mejor de mí y por ser la mejor madre del mundo; aquí está tu regalo.

A mi papá por apoyarme siempre, por confiar en mí y por quererme como sólo él sabe hacerlo.

A mi hermana querida, por ser un ejemplo para mí. A mi sobrinito.

A toda mi familia, por su preocupación y cariño.

A Susy, mi compañera de Tesis y mi amiga, por tanto esfuerzo que hicimos juntas para realizar este trabajo y por estar siempre ahí cuando lo necesité, te quiero mucho.

A mis mejores amigas de ahora y siempre, Arasay y Ariuska, que han estado ahí para mí, en los buenos y malos momentos de mi vida. Gracias por su cariño.

A mis compañeros de aula y amigos que siempre estuvieron junto a mí en las buenas y malas, especialmente a Ana, Jas, May, Yami, Lidy, Yury, Alain, Annier, a Alejandro por su eterno cariño y comprensión.

A mis amigos del Pre-universitario que gracias también a ellos estoy aquí ahora.

A Yunier por haber estado ahí siempre, por su cariño, y por su apoyo a lo largo de estos cinco años.

A mis tutores Heney y Maikel por su ayuda en la realización de este trabajo.

A todos mis profesores, en especial a Pedro Piñeiro y a Pascual por su apoyo.

A nuestro Comandante por darme la oportunidad de haber cursado mis estudios superiores en esta maravillosa Universidad, a la cual le estoy muy agradecida por las cosas nuevas que aprendí, por las experiencias maravillosas y por las nuevas amistades adquiridas.

Dania Durán Cutiño

A mi mamá por ser tan especial y estar siempre, quererme tanto, darme apoyo en los momentos más difíciles de mi vida, por confiar en mí y por ser la mejor madre del mundo.

A mi papá por ser un padre muy exigente, quererme tanto, confiar en mí, por aconsejarme en muchos momentos de mi vida y darme fuerzas para seguir adelante cuando lo he necesitado.

A mi hermanito del alma Reynier por ser mi ejemplo en todo momento y aconsejarme, gracias por ser el mejor hermano y amigo del universo, te quiero.

A mi hermana Anita que a pesar de la distancia siempre ha estado junto a mí y por ser tan paciente conmigo y saber entenderme.

A mi sobrinito y a mi hermana Fabi por quererme mucho.

A mi abuelita Olga por estar siempre amándonos y apoyándonos.

A toda mi familia, por preocuparse por mí en especial a mis tíos y primos.

A ti BBB por tu amor, cariño, apoyo y por estar siempre para mí.

A mi amiga, mejor dicho, hermana Adriana que siempre ha estado junto a mí en todo momento y ser tan especial.

A Danita, mi compañera de tesis y más que eso amiga por hacer tan buen equipo, pues juntas pudimos realizar esto.

A mi gran amiga Aineris por compartir tantas alegrías, músicas, fiestas, teatros, por estar en los buenos y malos momentos presente y darme apoyo.

A todos mis compañeros de aula y amigos en especial Digna, Lisandra, Mailin, Lidy, May, Yami, Jas, Ana, Osmar, Alián, Alain, Maikel, Yunier y a Alejandro por su comprensión y amor.

A todos aquellos que contribuyeron a mi formación de una u otra forma, en especial a Pascual y a Pedro Y. Piñeiro por confiar en mi y ayudarme siempre que lo he necesitado.

A mis tutores Heney y Maikel Yelandi por la ayuda brindada en este tiempo.

A nuestro Comandante en Jefe Fidel Castro Ruz por darme la posibilidad de haber estudiado en esta magnífica escuela.

Susana Gonzalez Espinosa

Dedicatoria

A mis padres queridos y a mi hermana, por ser mis guías, mis amigos incondicionales, mi razón de ser, por ser lo más grande que tengo en este mundo. Aunque toda una vida no alcanzará para recompensarlos, espero que alcance al menos para hacerlos sentir orgullosos.

Dania Durán Cutiño

Dedico este trabajo a mis padres y a mis hermanos personas extraordinarias, que sin ellos no seria lo que soy, pues me han guiado siempre, me han dado su apoyo y me han llenado de su cariño, amor y sensibilidad, gracias por todas esas cosas que me han dado; lo más preciado que tiene uno es la vida y esa también me la han dado ellos. Todo lo que haga es poco para retribuirlos pero espero alcance al menos para gratificarles y expresarles que todo lo que soy y tengo es gracias a ellos.

Susana Gonzalez Espinosa

Resumen

El presente trabajo se realizó con el propósito de elaborar una estrategia que contribuya a la mejora de la calidad en el sistema Administración Financiera del proyecto Registros y Notarías.

El cimiento de la calidad del software es el cumplimiento de los requisitos que solicita el cliente, de forma que este se sienta satisfecho con la creación del producto. Con este propósito, se le deben realizar pruebas para encontrar la mayor cantidad de defectos posibles y que este llegue a las manos del cliente con la calidad requerida.

Este trabajo se centra en la elaboración de una estrategia apoyada en la metodología RUP, en la que se definieron procesos que guían la realización de las pruebas mediante las técnicas de Caja Blanca y Caja Negra, de forma que estas se efectúen con la calidad que debe presentar un proyecto de esta índole. Se realizó una selección de los roles que deben intervenir en cada uno de los procesos, así como los artefactos necesarios para que estos se encuentren debidamente documentados, dando paso a cada una de las actividades a desarrollar.

La estrategia desarrollada fue aplicada en el módulo Requisiciones del sistema Administración Financiera del proyecto Registros y Notarías. Se realizó un análisis de los resultados reales y esperados con el propósito de corregir los defectos encontrados. Mediante la correcta aplicación de esta estrategia se garantiza que se realice la detección de errores en la fase correspondiente, garantizando que en la liberación del producto este se encuentre libre de defectos.

Palabras claves

Proceso de pruebas de software

Índice

Introducción

Uno de los problemas que afronta actualmente el mundo de la informática es la calidad del software. Desde la década del 70, este tema ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software, los cuales han realizado gran cantidad de investigaciones al respecto con dos objetivos fundamentales: ¿cómo obtener un software con calidad? y ¿cómo evaluar la calidad del software? (1)

Este proceso es de suma importancia ya que si se detectan problemas luego de elaborado el producto, no solo traería pérdidas monetarias, sino pérdidas de tiempo y esfuerzo, por lo que es imprescindible tener en cuenta durante todas las etapas del ciclo de vida del software, como es desarrollado este y cuales son las características del producto que se va obteniendo.

Actualmente el proceso de transformación y perfeccionamiento de las tecnologías ha traído consigo que se incremente cada día, la lista de países productores de software, donde la calidad de estos es imprescindible para la competencia y la inserción en el mercado. En este aspecto nuestro país esta dando un paso de avance, existen empresas cubanas dedicadas a la producción y comercialización de software como son Softel, Desoft y Segurmática. Con el mismo objetivo se ha definido como táctica, la creación de la UCI1 como centro de investigación y desarrollo de software. En la UCI se desarrollan diversos proyectos de software donde se hace necesario aplicar apropiadamente metodologías, procedimientos, estándares y emplear herramientas que permitan desarrollar un software con calidad.

Para ello la universidad creó la dirección de calidad central que ha realizado diversas funciones entre las que se encuentran la propuesta de diferentes políticas, estrategias y herramientas con el objetivo de que el proceso de calidad en los proyectos productivos sea efectivo.

A pesar de los esfuerzos realizados por la dirección de calidad, todavía el tema de las pruebas de software constituye un problema vigente ya que en los proyectos las mismas no se efectúan en la etapa adecuada de su desarrollo, lo cual es de suma importancia para que en el momento de la liberación del producto, no se detecten errores que traerían costosas pérdidas de tiempo y esfuerzo. Todo esto se puede evitar llevando organizadamente y en el momento preciso dicho proceso.

Uno de los proyectos de mayor importancia que se desarrollan actualmente en nuestra Universidad consiste en la realización de un software para la informatización de los Registros y Notarías de

[1] Universidad de las Ciencias Informáticas

Venezuela. Este proyecto está conformado por varios módulos en los cuales no se comprueba si el código fuente del producto posee errores lógicos y suposiciones incorrectas, permitiendo llegar a la fase de integración sin un alto grado de seguridad de que el código está funcionando correctamente. Además no se siguió la metodología propuesta ya que no se realiza una selección correcta de los roles, ni de las actividades que estos deben desempeñar. El proceso de pruebas también se ha visto afectado por la ausencia de herramientas para su automatización. Estos detalles son muy necesarios y nos llevarían a obtener un producto con mayor calidad.

Debido a esta situación, se ha definido el siguiente **problema**:

¿Cómo mejorar el proceso de pruebas en el proyecto Registros y Notarías para obtener un producto con un número reducido de defectos?

Nuestro **objeto de estudio** es: Calidad de software.

El **campo de acción** que se propone es: Proceso de pruebas en el proyecto Registros y Notarías.

Para dar solución a la problemática planteada, se trazó el siguiente **objetivo general**:

Elaborar una estrategia para la aplicación de las técnicas de prueba de Caja Blanca y Caja Negra para el proyecto Registros y Notarías.

La **hipótesis** que se plantea es: La aplicación de una estrategia para el proceso de pruebas en el proyecto Registros y Notarías garantiza la obtención de un producto con un número reducido de defectos.

Para poder desarrollar una estrategia que nos permita llevar a cabo un buen proceso de pruebas, aplicando las técnicas de Caja Blanca y Caja Negra al proyecto Registros y Notarías, es necesario trazar **tareas de investigación** que guíen el camino hacia el cumplimiento de los objetivos definidos anteriormente y así obtener los resultados esperados:

- Realización de un estudio sobre el estado del arte de las técnicas de prueba de software.
- Análisis de los lineamientos establecidos por la dirección de Calidad de la Universidad de las Ciencias Informáticas.
- Análisis de las mejores prácticas para la automatización del proceso de pruebas.
- Selección de las herramientas para la automatización de las pruebas.
- Elaboración del marco de procesos que formarán parte de la estrategia.
- Aplicación de la estrategia en un módulo del proyecto Registros y Notarías.

- Comparación de los resultados esperados y reales que se obtienen de la ejecución de las pruebas.

El trabajo consta de los siguientes capítulos:

El capitulo 1 se centra en el fundamento teórico del objeto de estudio; se abordan aspectos importantes como la implicación que tiene la prueba en la calidad del software, se define su concepto, sus objetivos y principios, así como la descripción del proceso de prueba. Se realiza además una descripción de las técnicas de las pruebas de Caja Blanca y Caja Negra existentes, así como las herramientas que existen para su automatización.

El capítulo 2 se centra en definir los procesos que formarán parte de la estrategia. Se planifican los recursos y métodos para viabilizar la ejecución de las pruebas y se realiza la selección de las herramientas adecuadas para llevar a cabo la automatización de estas.

En el capítulo 3 se lleva a cabo la estrategia definida en el módulo Requisiciones del sistema Administración Financiera del proyecto Registros y Notarías. Se comparan los resultados esperados con los reales obtenidos en la ejecución de las pruebas.

Capítulo 1: Fundamentación Teórica.

1.1 Introducción

En el presente capítulo se hace un breve análisis de la calidad del software y de los conceptos de prueba de software dados por varios autores. Se dan a conocer los objetivos de las pruebas, principios que la rigen y una descripción del proceso. Se abordan además las pruebas de Caja Blanca y Caja Negra, así como los distintos niveles de prueba. Se realiza un estudio sobre el trato que le dan las metodologías RUP y XP a las pruebas, y se hace un estudio del estado del arte a nivel nacional, internacional y en la Universidad de Ciencias Informáticas para saber cuáles son las tendencias fundamentales y el nivel de importancia que se le da a dicho proceso.

1.2 Calidad del software

Lograr un alto nivel de calidad es el objetivo de la mayoría de las organizaciones que desarrollan software. La administración de la calidad utiliza procedimientos y estándares durante el desarrollo de software, además del correspondiente proceso que verifica que el personal siga estos estándares.

Actualmente es extraño ver alguna empresa que no este interesada en la calidad, la gran mayoría que se dedica al desarrollo de software implementa alguna metodología y/o tiene un equipo dedicado a verificar la calidad de sus productos.

De acuerdo a la terminología de la IEEE, la calidad de un sistema, componente o proceso de desarrollo de software, se obtiene en función del cumplimiento de los requerimientos iniciales especificados por el cliente o usuario final.

La calidad del software continúa siendo un problema actual ya que la misma afecta tanto a los productores de software como a los clientes. Con el aumento de la informatización a escala mundial se ha incrementado de forma notable la producción de software, lo que trae consigo que se priorice más el proceso para obtener un software con calidad.

Debido a la importancia que esto ha despertado en los últimos años, la Conferencia Internacional de la Ingeniería de Software del año 2002 (ICSE 2002) se centró en los aspectos de calidad. En esta conferencia se concluyó que los criterios de calidad más importantes son la fiabilidad, usabilidad, seguridad, disponibilidad, escalabilidad, mantenibilidad.

Otro factor que influye en este proceso es la competencia existente, pues hay instituciones que validan que el software tiene calidad, es decir, hay empresas dedicadas solamente al aseguramiento de la calidad, lo que es un aval para el cliente saber que el producto que esta adquiriendo fue certificado por una de estas empresas.

Actualmente existen estándares para el control de la calidad y para su proceso de mejora continua entre los que se encuentran: CMMI, ISO 9001: 2000, ISO 9004: 2000, ISO 9001: 2005.

CMMI (Capability Maturity Model Integration) es un modelo para la mejora de procesos que proporciona a las organizaciones los elementos esenciales para procesos eficaces. CMMI se desarrolló para facilitar y simplificar la adopción de varios modelos de forma simultánea integrando a sus predecesores: CMM-SW (CMM for Software), SE-CMM (Systems Engineering Capability Maturity Model), IPD-CMM (Integrated Product Development). CMMI es el sucesor de CMM que fue desarrollado desde 1987 hasta 1997. En el 2002, SW-CMM fue integrado y relevado por el nuevo modelo CMMI, se lanzo la versión 1.1, luego en Agosto del 2006 siguió la versión 1.2. Las mejores prácticas CMMI se publican en los documentos llamados modelos. En la actualidad hay dos áreas de interés cubiertas por los modelos de CMMI: Desarrollo (DEV-CMMI) y Adquisición (ACQ-CMMI), estos son los dos modelos de la versión 1.2 disponibles. Independientemente del modelo que opta una organización, las prácticas CMMI deben adaptarse a cada organización en función de sus objetivos de negocio.

En el equipo de desarrollo de CMMI había defensores de ambos tipos de representaciones. El resultado fue la publicación del modelo con dos representaciones: continua y escalonada. Son equivalentes, y cada organización puede optar por adoptar la que se adapte a sus características y prioridades de mejora. La visión continua de una organización mostrará la representación de nivel de capacidad de cada una de las áreas de proceso del modelo (2). La visión escalonada definirá a la organización dándole en su conjunto un nivel de madurez del 1 al 5.

La familia de normas ISO 9000 son normas de calidad establecidas por la Organización Internacional para la Estandarización (ISO) que se pueden aplicar en cualquier tipo de organización. Se componen de estándares y guías relacionados con sistemas de gestión y de herramientas específicas como los métodos de auditoría.

Su implantación en estas organizaciones, aunque supone un duro trabajo, ofrece una gran cantidad de ventajas para sus empresas. Los principales beneficios son: reducción de rechazos e incidencias en la producción o prestación del servicio, aumento de la productividad, mayor compromiso con los requisitos del cliente y mejora continua (3).

La ISO 9001 es una norma internacional que se aplica a los sistemas de gestión de calidad (SGC) y que se centra en todos los elementos de administración de calidad con los que una empresa debe contar para tener un sistema efectivo que le permita administrar y mejorar la calidad de sus productos o servicios. Las principales normas de la familia ISO son: ISO 9001:2000 (Sistemas de Gestión de la Calidad – Requisitos) e ISO 9004:2000 (Sistemas de Gestión de la Calidad - Guía de mejoras del funcionamiento) (4).

Lamentablemente los desarrolladores le restan importancia al proceso que se lleva a cabo para obtener un software con calidad, en muchas ocasiones se omiten la realización de pruebas en determinadas fases, lo que trae consecuencias nefastas y el costo del software puede incrementarse a tal punto que produzca grandes pérdidas.

Cuando se obtiene un software con la calidad requerida se consigue reducir el número de errores o erradicarlos completamente. Un pilar fundamental para obtener un producto con calidad lo constituyen las pruebas de software, las cuales se tratarán en el siguiente epígrafe.

1.3 Pruebas de software. Definiciones.

En la actualidad es muy importante obtener la medida de calidad con la que cuenta un software, debido a la competitividad que existe en el mercado de estos productos y al actual crecimiento que ha tenido el número de empresas que se desarrollan en este campo.

Para determinar la calidad de un producto de software se deben efectuar medidas y desarrollar actividades que permitan comprobar el grado de cumplimiento de las especificaciones iniciales del sistema, y es aquí donde las pruebas de software desempeñan un papel fundamental. Estas verifican el desarrollo que va alcanzando el producto durante todas sus etapas, identificando posibles fallos de implementación, calidad, o usabilidad de un programa.

Existen varias definiciones dadas por diversos autores sobre lo que es prueba de software, se hará referencia a algunas de estas a continuación:

Pruebas de software según (5) es el proceso de evaluación de un producto desde un punto de vista crítico, donde el "probador" (persona que realiza las pruebas) somete al producto a una serie de acciones indagadoras, y el producto responde con su comportamiento como reacción. Es necesario probar los nuevos programas en un entorno de pruebas separado físicamente del de producción.

Se llama prueba de software, según (6) al proceso en el que se ejecuta un sistema con el objetivo de detectar fallos.

La IEEE (7) plantea que la prueba es la actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan y almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.

Según RUP la prueba es una disciplina en el proceso de ingeniería de software cuyo objetivo es integrar y poner a prueba el sistema.

Estos son algunos de los principales conceptos de prueba manejados actualmente en el mundo del software. Todos ellos apuntan a la significativa importancia de este proceso durante todo el desarrollo de un producto.

Se considera que la prueba de software es una etapa imprescindible durante todo el proceso de desarrollo, pues una vez que se genera código fuente, el software debe ser probado para descubrir y corregir el máximo de errores posibles antes de su entrega al cliente. Al diseñar una serie de casos de pruebas que tengan una gran probabilidad de encontrar errores se cumple el objetivo fundamental. Para desempeñar el mismo se llevan a cabo técnicas de pruebas del software, las que facilitan una guía sistemática para diseñar pruebas que comprueben la lógica interna de los componentes software y el grado de cumplimiento de los requisitos funcionales.

Es necesario conocer los objetivos de las pruebas pues estos guían el proceso. A continuación se describen los mismos.

1.3.1 Objetivos de las pruebas

Glen Myers (8) expresó en “The Art of Software Testing” que los objetivos de las pruebas concretamente pueden ser:

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Mientras que Bill Hetzel (9), en “The Complete Guide To Software Testing”, planteó como objetivos de las pruebas de software:

- Planificar las pruebas necesarias en cada iteración.
- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, creando los procedimientos de prueba que especifican cómo realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas.
- Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados.

El cumplimiento de estos objetivos es de gran importancia ya que este proceso posibilita no solo la detección de errores en el código sino la documentación necesaria que facilita que este código pueda ser reutilizado. Un buen resultado de la implementación no solo depende de la codificación sino también de las pruebas.

La realización de un proceso adecuado facilita que disminuyan los riesgos asociados y además que se eleve el nivel de calidad del producto.

Una vez conocidos los objetivos de las pruebas, se deben seguir una serie de principios para que el proceso tenga calidad, éstos se explican a continuación.

1.3.2 Principios de las pruebas

Para el diseño de casos de prueba efectivos, un ingeniero de software deberá conocer los principios básicos que guían las pruebas del software. En el libro “201 Principles of Software Development”, Davis A. (10) define entre otros principios:

- A todas las pruebas se les debería poder hacer un seguimiento hasta los requisitos del cliente.
- Las pruebas deberían planificarse mucho antes de que empiecen.
- El principio de Pareto (8) es aplicable a la prueba del software.
- Las pruebas deberían empezar por “lo pequeño” y progresar hacia “lo grande”.
- No son posibles las pruebas exhaustivas.
- Para ser más eficaces, las pruebas deberían ser realizadas por un equipo independiente.

Se considera que para obtener un buen caso de prueba se debe tener en cuenta también, que no se hagan pruebas que tengan el mismo propósito, es decir, redundantes, ya que se pierde tiempo en el diseño y ejecución de las mismas.

Una vez completado el modelo de requisitos, es recomendable comenzar con la planificación de las pruebas; y luego de consolidar el modelo de diseño, podemos comenzar con la definición detallada de los casos de prueba. Por tanto, se puede planificar y diseñar todas las pruebas antes de generar ningún código.

La realización de un proceso adecuado facilitará que una vez que el producto llegue a manos del cliente o usuario final tendrá la menor cantidad de errores logrando con esto su satisfacción.

Es necesario seguir un conjunto de acciones para la realización de las pruebas, de esta forma se organizaría más el trabajo y se realizaría de forma más efectiva, posibilitando el éxito del objetivo propuesto. Estas acciones se explican a continuación.

1.3.3 Procesos de prueba

Un proceso según (11) es un conjunto de acciones que se realizan de forma secuencial y que en su conjunto proporcionan valor añadido a las entradas, con el fin de producir unas salidas que satisfagan las necesidades de los clientes.

El proceso de pruebas debe ser realizado de una forma muy organizada para la obtención de un mejor resultado. Entre estos procesos se encuentran la planificación, el diseño de casos de prueba, la implementación, ejecución y los resultados; tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software.

- Planificación de las pruebas: En esta fase es importante describir una estrategia de prueba, estimar los requisitos para el esfuerzo de la prueba, ejemplo, los recursos humanos y sistemas necesarios. Se debe tener conocimiento de las herramientas empleadas actualmente en el mercado y se definen los roles y responsabilidades en los equipos. Es necesario determinar el ámbito y alcance de un plan de pruebas y su proceso de ejecución.
- Diseño de las pruebas: La facilidad de prueba sucede como el resultado de un buen diseño. El diseño de datos, de la arquitectura, de las interfaces y de los componentes de

detalles pueden facilitar la prueba o hacerla más fácil. Por lo que se recomienda seleccionar las técnicas adecuadas para el diseño de casos y procedimientos de prueba.

- Implementación de las pruebas: Selección de las herramientas necesarias para la automatización de las pruebas.
- Ejecución de las pruebas: Luego de obtener los casos de pruebas diseñados se procede a ejecutar los mismos para la detección de errores.
- Evaluación de los resultados: Se verifica la correspondencia que existe entre los resultados esperados y reales y se obtienen estadísticas de los errores.

1.4. Niveles de Prueba

Para llevar a cabo el proceso de prueba durante el desarrollo del software, existen niveles de prueba que permiten probar el producto desde la menor unidad creada hasta que se ha finalizado la construcción del mismo; cada uno se realiza en determinado momento del ciclo de desarrollo del software. Estos niveles son los siguientes:

Pruebas de unidad:

La prueba de unidad se centra en la revisión de los módulos, siendo la escala más pequeña, donde se prueban todos los componentes implementados como unidades individuales. Para llevar a cabo esta prueba se emplea fundamentalmente la técnica de Caja Blanca que es la que verifica que el código funcione correctamente, sin omitir la técnica de Caja Negra que es la encargada de comprobar el funcionamiento de la unidad externamente. Se prueban todos los caminos posibles con el objetivo de descubrir errores en la pequeña unidad que se esta probando.

Pruebas de integración:

La prueba de integración consiste en probar los módulos individuales combinados; después de haber probado cada unidad del software por separado se pasa a probar estas unidades pero integradas, donde se comprueba si estas partes juntas funcionan correctamente; es decir, se verifica su buen funcionamiento. Esta es una técnica donde se construye la estructura del programa y a su vez se llevan a cabo las pruebas para detectar errores asociados con la integración.

Pruebas de sistema:

La prueba del sistema constituye el proceso de un sistema integrado de hardware y software que permite comprobar que los requisitos funcionales se cumplen, así como el buen funcionamiento y rendimiento del software conjuntamente con la parte hardware que implica dicho software. Permite una adecuación de la documentación de usuario así como la ejecución y rendimiento en condiciones límites y de sobre carga. Las pruebas de sistema persiguen la integración de cada módulo, además de buscar las discrepancias entre el sistema, especificaciones y su documentación.

Pruebas de aceptación:

La prueba de aceptación es una prueba determinante para la justificación de las ventajas que ofrece el empleo del producto desarrollado. En esta prueba se evalúa el grado de calidad del software, y es realizada por un grupo de usuarios finales o los clientes, para asegurarse de que el sistema cumple con los requisitos planteados.

Prueba alfa:

Las pruebas alfa consisten en llevar al cliente al entorno de desarrollo para probar el sistema. De esta forma el cliente trabaja en un entorno controlado y siempre tiene un experto a mano para ayudarle a usar el sistema y para analizar los resultados. Al final de la prueba el encargado del proyecto registra las sugerencias, errores y mejoras posibles para el sistema.

Prueba beta:

Las pruebas beta se realizan luego de las pruebas alfa, y se desarrollan en el entorno del cliente, es decir, en el lugar de trabajo para el que fue diseñado el sistema. Generalmente en este tipo de pruebas el encargado del desarrollo no se encuentra presente. De esta forma el cliente puede probar el producto a su manera, tratando siempre de encontrarle fallos, los que registra e informa más tarde al desarrollador.

Prueba de regresión:

Las pruebas de regresión consisten en verificar que el software sigue cumpliendo las especificaciones y requisitos después de cada versión. Son realizadas por desarrolladores y analistas de la aplicación. Pueden ser ejecutadas en el nivel de integración, para minimizar costos.

1.5. Pruebas de Caja Blanca.

Cuando se implementa un software se pretende garantizar la obtención de un producto con calidad, para lo que resulta recomendable comprobar que el código que hemos escrito funciona correctamente. Con este propósito se implementan pruebas que verifican que el programa genera los resultados que realmente esperamos. Las pruebas realizadas al código son las llamadas pruebas de Caja Blanca, y se basan en un examen minucioso de los detalles procedimentales, logrando examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. Estas no se deben confundir con las pruebas informales que realiza el programador mientras está desarrollando el módulo.

Las pruebas de Caja Blanca han tomado un lugar muy importante en el desarrollo de sistemas de cualquier tipo, tanto que sin esta prueba un sistema desarrollado carecería de garantías y credibilidad en sus resultados.

Estas pruebas constituyen un método mediante el cual el ingeniero del software puede obtener casos de prueba que garanticen:

- Ejercitar por lo menos una vez todos los caminos independientes de cada módulo.
- Ejercitar todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Comprobar los ciclos en sus límites y con sus límites operacionales.
- Ejercitar las estructuras de datos internas para asegurar su validez (6).

El principal factor que se debe considerar al inicio de las pruebas es el tamaño del módulo a probar, se debe considerar si el tamaño del módulo permitirá probar adecuadamente toda su funcionalidad de manera sencilla y rápida. También es importante separar los módulos de acuerdo a su funcionalidad, si los módulos son muy grandes y contienen muchas funcionalidades, estos se volverán más complejos de probar y al encontrar algún error será más difícil ubicar la funcionalidad defectuosa y corregirla. Al hacer esta labor el analista de pruebas podrá recomendar que un módulo muy complejo sea separado en 2 o 3 módulos más sencillos.

Cada método de una clase puede ser sometido a pruebas para determinar si está correctamente implementado de acuerdo con la especificación que contiene el modelo funcional. Algunos métodos son sencillos, los cuales no conllevan la realización de muchas pruebas.

En la Figura 1 se muestra lo que se considera una representación clásica de pruebas de Caja Blanca. En este tipo de pruebas el cubo representaría un sistema en donde se pueden observar

los diversos componentes que forman parte del mismo, cada uno de estos componentes debe ser probado en su totalidad (óvalos), también sus interfaces o comunicaciones con los demás componentes (flechas), este tipo de pruebas también son llamadas pruebas de caja de cristal ya que este último termino representa mejor el tipo de pruebas.

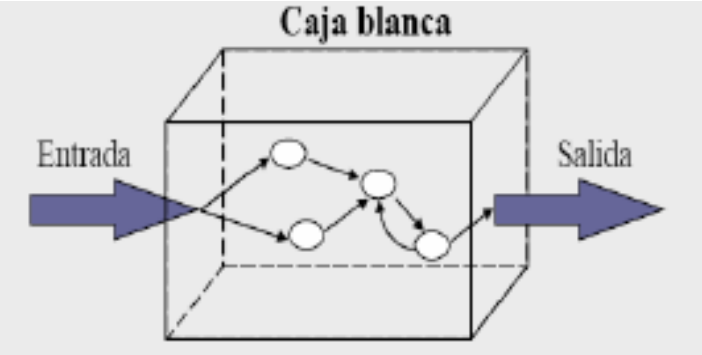


Figura 1: Prueba de Caja Blanca

Cuando se aplican estas pruebas se debe tener en cuenta los siguientes aspectos:

- Los datos de entrada son conocidos por el Probador o Analista de Pruebas y deben ser preparados con minuciosidad, ya que el resultado de las pruebas depende de estos.
- Se debe conocer qué componentes interactúan en cada caso de prueba y qué resultados debe devolver el componente según los datos de entrada utilizados en la prueba.
- Finalmente se deben comparar los datos obtenidos en la prueba con los datos esperados, si son idénticos podemos decir que el módulo superó la prueba y empezamos con la siguiente.

Para lograr un buen resultado a partir de la aplicación de las pruebas de Caja Blanca, no se debe esperar a que esté todo el código escrito para comenzar a probar. Estas pruebas deben irse realizando según se va generando el código, facilitando con esto que los errores se detecten lo antes posible. Es recomendable que estas pruebas sean diseñadas y aplicadas por una persona distinta a la que escribe el código; siendo esta la única forma de no ser "comprensivo con los fallos".

Para aplicar las pruebas de Caja Blanca existen diferentes métodos. El empleo de estos permite encontrar el conjunto de caminos existentes en una porción de código; ejercitar las condiciones lógicas comprendidas en un módulo; validar las construcciones de los ciclos; entre otras funcionalidades. Para obtener un mayor conocimiento acerca de estos métodos se hace un breve análisis de los mismos a continuación.

1.5.1. Métodos de prueba de Caja Blanca

Existen varios métodos de prueba de Caja Blanca, entre los que se encuentran la prueba del camino básico, prueba de condición, prueba de bucles y prueba de flujo de datos. Estos métodos se explican a continuación.

Prueba del camino básico: Es una técnica de prueba de Caja Blanca propuesta inicialmente por Tom McCabe. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. En este tipo de pruebas se elaboran grafos de flujo, se determina su complejidad ciclomática para obtener el conjunto básico de caminos linealmente independientes y de esta forma los casos de prueba del conjunto básico con el objetivo de ejecutar al menos una vez cada sentencia del programa.

Según estudios realizados (12), (6), los pasos del método son:

1. Dibujar G, el grafo del método de la clase.

Los tres componentes fundamentales de un grafo de flujo son: los nodos, las aristas y las regiones. Se define como nodo a cada círculo representado en el Grafo de Flujo, el cual representa una o más secuencias procedimentales. Las aristas son las flechas del grafo y representan el flujo de control, esta debe terminar en un nodo. Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más.

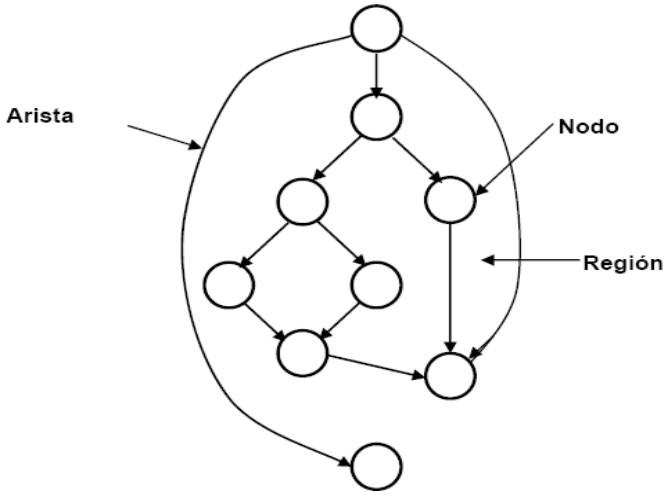


Figura 2: Componentes del grafo

Para construir el grafo se cuenta con la siguiente notación:

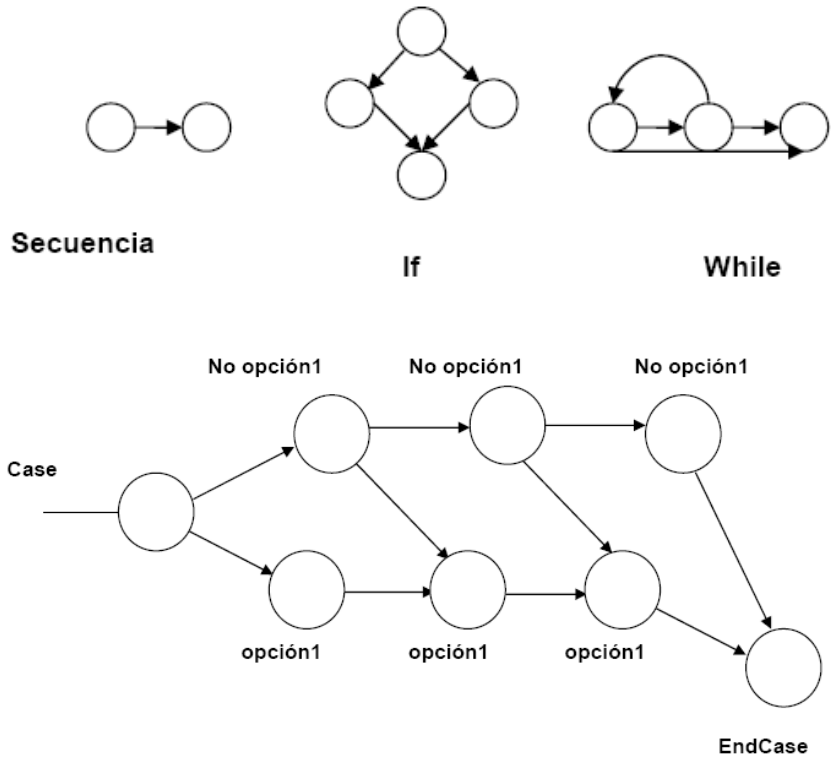


Figura 2: Notación para las instrucciones: Secuenciales, If, While y Case

2. Determinar el número ciclomático del grafo, V (G).

   Existen diferentes formas de calcular la complejidad ciclomática. Esta se puede definir como V (G) = A - N + 2, donde A es el número de aristas del grafo y N es el número de nodos. También se puede obtener de la forma V (G) = P + 1, siendo P el número de nodos predicado contenido en el grafo G. Por último, el número de regiones del grafo coincide con la complejidad ciclomática.

3. Construir una secuencia de V(G) caminos linealmente independientes en G, recursivamente:

   (a) El primer camino es cualquiera de los caminos mínimos del nodo origen del grafo a uno de los nodos de terminación del grafo. Se inicializa A (G), el conjunto de aristas en la secuencia linealmente independiente de G con todas aquellas aristas que forman este primer camino.

   (b) El próximo camino se forma agregando un nuevo camino entre el origen y una terminación de G. Este nuevo camino debe agregar el mínimo número posible de aristas nuevas a A, (pero que agregue por lo menos una arista nueva).

4. Preparar un caso de prueba por camino hallado en el paso anterior.

   (a) Determinar los datos a proporcionar como entrada para ejecutar el camino hallado.

   (b) Usando la especificación funcional del método, indicar cuál es el resultado esperado.

A partir de esto, podemos establecer que la complejidad ciclomática es una métrica del software que proporciona una medición de la complejidad lógica de un programa.

Prueba de condición: Ejercita las condiciones lógicas contenidas en el módulo de un programa para descubrir errores de operadores lógicos, relacionales, paréntesis lógicos o expresiones aritméticas. Esta técnica permite además detectar otros errores dentro de dicho programa.

Si se tiene una expresión relacional de la forma:

E1 <operador relacional> E2

Se requieren tres pruebas:

Prueba 1: E1 > E2 Prueba 2: E1 < E2

Si la expresión es de la forma:

If Condicion1 or Condicion2 then B

Se considera la realización de cuatro pruebas:

Prueba 1: Condicion1 = true y Condicion2 = false
Prueba 2: Condicion1 = false y Condicion2 = true
Prueba 3: Condicion1 = false y Condicion2 = false
Prueba 4: Condicion1 = true y Condicion2 = true

Existen algunas estrategias de pruebas de condiciones que permiten dar una orientación para la generación de pruebas adicionales del programa, entre otras se pueden mencionar:

- Prueba de ramificación: Se centra en la necesidad de ejecutar una sentencia determinada, al menos una vez, en sus vertientes verdadera y falsa.
- Prueba del dominio: Requiere de la realización de tres o cuatro pruebas para una expresión relacional.

- BRO5: Esta se basa en las dos técnicas anteriores garantizando que se detecten errores de ramificaciones y de operadores relacionales en una condición dada.

Tipos de errores que pueden aparecer en una condición:

- Existe un error en un operador lógico
- Existe un error en un paréntesis lógico
- Existe un error en un operador relacional
- Existe un error en una expresión aritmética

Prueba de bucles: Con el uso de esta técnica se validan las construcciones de bucles con diferentes grados de complejidad (6):

- Bucles simples: Se realizan una serie de pruebas donde “n” es el número máximo de pasos permitidos por el bucle.
  - Pasar por alto totalmente el bucle.
  - Pasar una sola vez por el bucle.
  - Pasar dos veces por el bucle.
  - Hacer m pasos por el bucle, con m<n.
  - Hacer n-1, n y n+1 pasos por el bucle.

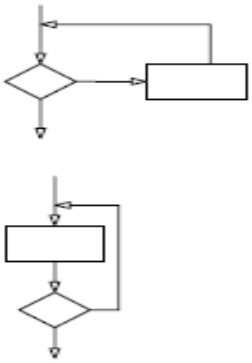


Figura 3: Bucle Simple

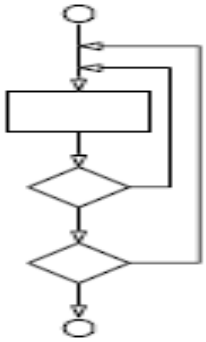


Figura 4: Bucle Anidado

- Bucles anidados: Para estos bucles se comienza la prueba por el bucle más interior.

- Comenzar por el bucle más interior. Establecer o configurar los demás bucles con sus valores mínimos.
- Llevar a cabo las pruebas de bucles simples para el bucle más interior, mientras se mantienen los parámetros de iteración (por ejemplo, contador del bucle) de los bucles externos en sus valores mínimos. Añadir otras pruebas para valores fuera de rango o excluidos.
- Progresar hacia fuera, llevando a cabo pruebas para el siguiente bucle, pero manteniendo todos los bucles externos en sus valores mínimos y los demás bucles anidados en sus valores típicos.
- Continuar hasta que se hayan probado todos los bucles.

- Bucles concatenados: Estos se pueden probar mediante el enfoque de los bucles simples si los bucles concatenados son independientes, en caso contrario, se usa el enfoque de los bucles anidados.

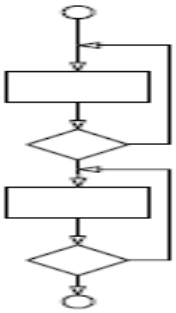


Figura 5: Bucle concatenado

- Bucles no estructurados: Este tipo debe rediseñarse para que se ajusten a las construcciones de programación estructurada.

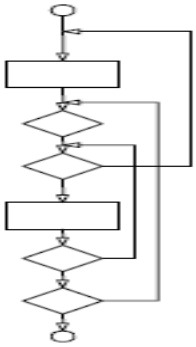


Figura 6: Bucle no estructurado

Prueba de flujo de datos: En este tipo de pruebas se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

La aplicación de los métodos de pruebas de Caja Blanca descritos anteriormente proporciona múltiples ventajas que llevarán al equipo de desarrollo a obtener un software con mayor calidad. A continuación se describen las ventajas de estas pruebas.

1.5.2. Ventajas de las pruebas de Caja Blanca

Las pruebas de Caja Blanca, según (13), proveen cinco ventajas básicas:

- Fomentan el cambio: Las pruebas de Caja Blanca facilitan que el programador cambie el código para mejorar su estructura. Esto permite hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
- Simplifican la integración: Las pruebas de Caja Blanca permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente.
- Documentan el código: Las propias pruebas son documentación del código y permiten ver cómo utilizarlo.
- Separación de la interfaz y la implementación.
- Los errores están más acotados y son más fáciles de localizar.

Se considera que las pruebas de Caja Blanca tienen una importancia trascendental en el ciclo de pruebas de un producto. Estas hacen que el programador se vuelva más consciente de sus decisiones de implementación; posibilitan la optimización del código con diferentes técnicas y acercamientos y consiguen encontrar defectos u operaciones que podrían proporcionar problemas futuros.

1.6. Pruebas de Caja Negra.

Las pruebas de Caja Negra (Figura 8) se centran en los requisitos funcionales, permitiendo al ingeniero del software derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Estas pruebas se llevan a cabo sobre la interfaz

del software y son completamente indiferentes al comportamiento interno y la estructura del programa. Los casos de prueba de Caja Negra pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, que se produce una salida correcta, y que se mantiene la integridad de la información externa.

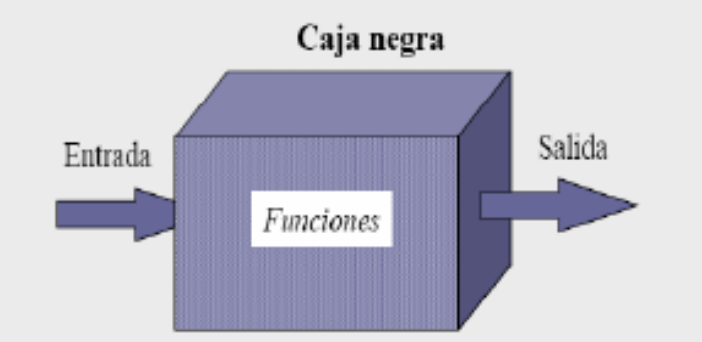


Figura 7: Prueba de Caja Negra

Estas pruebas tienen también como meta determinar la eficiencia del programa desde el desempeño en el equipo, el tiempo de retardo de las salidas hasta el nivel de recuperación del sistema luego de fallas o caídas, sean estas producidas por manejo incorrecto de datos, equipo, o producidas externamente.

Los casos de prueba de Caja Negra según varios autores (12) (14) pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta, y
- La integridad de la información externa se mantiene.

La prueba de Caja Negra intenta encontrar errores de los siguientes tipos:

- Funciones incorrectas o inexistentes.
- Errores relativos a las interfaces.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores debidos al rendimiento.
- Errores de inicialización o terminación.

Existen diferentes métodos que se utilizan en la aplicación de las pruebas de Caja Negra. El empleo de estos permite dado una serie de valores de entrada, obtener un conjunto de valores de

salida y además posibilita realizar una comparación entre varias versiones con los mismos datos de entrada, para poder verificar que las salidas sean las mismas. A continuación se hace un breve análisis de estos métodos.

1.6.1. Métodos de prueba de Caja Negra

Existen diferentes métodos de prueba de Caja Negra, algunos de estos se mencionan a continuación.

- Clases de equivalencia
- Análisis de valores límites
- Prueba de comparación
- Tabla Ortogonal
- Grafo Causa-Efecto

Clases de equivalencia: Esta técnica, conocida también como Partición equivalente, utiliza las clases de equivalencia para el diseño de los casos de prueba, es decir, la entrada de una serie de datos válidos y no válidos, cubriendo en cada caso el máximo número de entradas. Define casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Una condición de entrada puede ser un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

Reglas para la identificación de las clases de equivalencia.

- Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.

- Si una condición de entrada es lógica, se define una clase de equivalencia valida y una no válida.(6)

Análisis de valores límites: Esta técnica de prueba complementa a la partición equivalente. En lugar de centrarse solamente en las condiciones de entrada, este obtiene también casos de prueba para el campo de salida.

Reglas para el análisis de valores límites:

- Si una condición de entrada específica un rango delimitado por los valores a y b, se deben diseñar casos de prueba para los valores a y b, y para los valores justo por debajo y justo por encima de a y b, respectivamente.
- Si una condición de entrada específica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximo y mínimo, uno más el máximo y uno menos el mínimo.
- Aplicar las directrices anteriores a las condiciones de salida.
- Si las estructuras de datos internas tienen límites preestablecidos hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.(6)

Prueba de comparación: Esta técnica se utiliza en situaciones críticas en las que el software debe ser sumamente fiable. En ese tipo de aplicaciones, varios equipos separados desarrollan versiones diferentes de una aplicación, usando las mismas especificaciones. En esas situaciones se deben probar todas las versiones con los mismos datos de prueba, para proporcionar que todas arrojan los mismos resultados. Luego se ejecutan todas las versiones en paralelo y se hace una comparación en tiempo real de los resultados, para garantizar la consistencia.

Es aconsejable que aunque se vaya a distribuir una sola versión se desarrollen varias versiones de software independiente, ya que estas son la base de la técnica de prueba mano a mano o de comparación. A cada versión se le facilita como entrada los casos de prueba diseñados mediante alguna otra técnica como bien pudiera ser la partición de equivalencia. Si todas las salidas de las diferentes versiones son idénticas entonces se asume que las mismas son correctas sino se investiga cual fue la responsable de la diferencia para de esta forma determinar el defecto en una o varias de ellas.

La Prueba de la tabla ortogonal puede aplicarse a problemas en el que el dominio de entrada es relativamente pequeño pero demasiado grande para posibilitar pruebas exhaustivas. Las pruebas del Grafo Causa – Efecto consisten en crear un grafo a partir de las especificaciones, y seleccionar

suficientes casos de prueba como para asegurar la cobertura del grafo. En este caso se le llama causas a las características de los datos de entrada y los efectos son las clases de salidas que puede proporcionar el programa. Para esta investigación estas pruebas no son relevantes.

1.7. Metodologías de Desarrollo de Software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software. Estas van indicando paso a paso todas las actividades a realizar para lograr el producto deseado, indicando además, qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener cada una. Detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.

Si no se emplea una metodología para desarrollar un producto de software pueden surgir los siguientes problemas:

- Resultados impredecibles.
- Detección tardía de errores.
- La introducción de nuevas herramientas afectará perjudicialmente al proceso.
- Cambios de organización también afectarán al proceso.
- Resultados distintos con nuevas clases de productos.

Se puede plantear entonces que el uso de las metodologías constituye un paso significativo para la elaboración de un producto de software. Actualmente podemos encontrar diversas metodologías con diferentes enfoques: metodologías tradicionales como RUP y MSF y metodologías ágiles como XP, SCRUM, Crystal, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Feature -Driven Development (FDD), y Lean Development (LD).

A continuación se hace un estudio del tratamiento particular que le dan las metodologías RUP y XP a las pruebas de software.

1.7.1. RUP. Proceso Unificado

Rational Unified Process (RUP) es la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Tiene tres características fundamentales: Define el proceso como iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura. RUP ha agrupado las actividades en grupos lógicos definiendo 9 flujos de trabajo principales y en su ciclo de vida cuenta con cuatro fases: inicio, elaboración, construcción y transición, que guían cada uno de los flujos, como muestra la Figura 9.

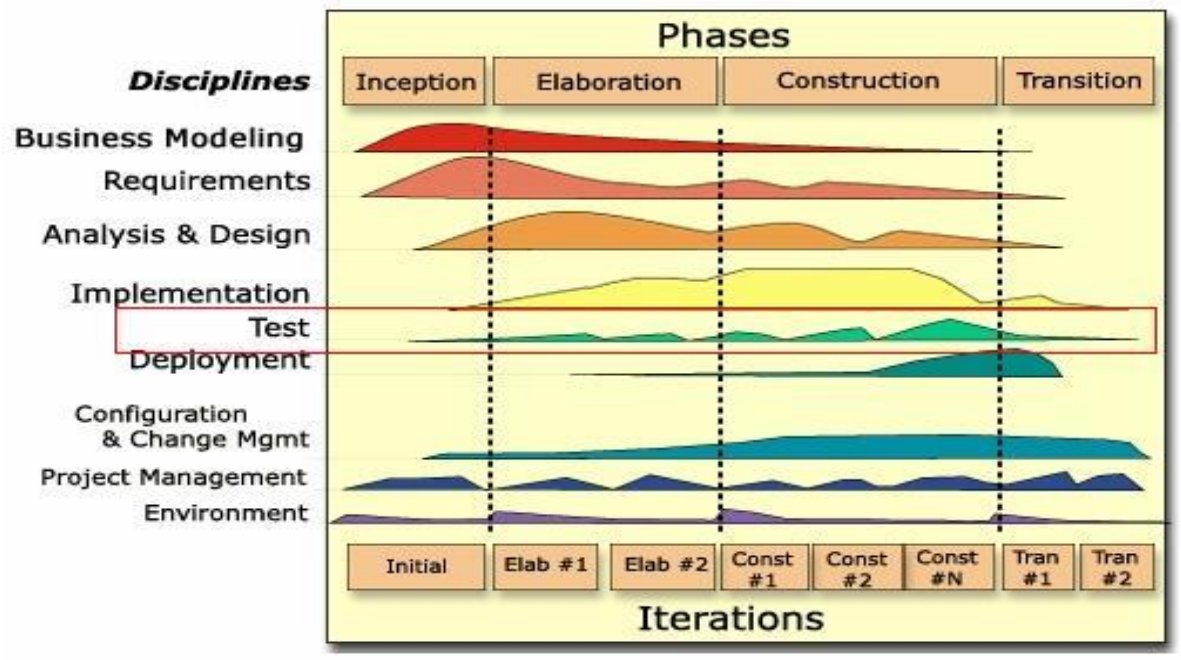


Figura 8: Fases y Flujos de trabajo del RUP

"La etapa de prueba es el flujo de trabajo fundamental cuyo propósito general es comprobar el resultado de la implementación mediante las pruebas de cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema que van a ser entregadas a terceras partes" (15). Es aconsejable que las pruebas se realicen desde el mismo momento en que comienza el desarrollo y continúe hasta que finaliza.

La metodología RUP propone diferentes roles que tienen como propósito el desarrollo de las pruebas de software. Estos son:

- Administrador de pruebas: Es el encargado del éxito y la planificación de las pruebas.
- Analista de pruebas: Es responsable de identificar y definir las pruebas, así como supervisar el progreso y los resultados de las mismas.

- Diseñador de pruebas: Es responsable de identificar las técnicas y herramientas necesarias, así como definir y estructurar los elementos de prueba.
- Probador: Es el encargado de ejecutar las pruebas, analizando los resultados obtenidos.

El Proceso Unificado de Desarrollo de Software plantea 7 artefactos fundamentales para el flujo de trabajo de prueba los cuales deben ser generados dentro de esta fase, estos son:

- Modelo de Prueba
- Caso de Prueba
- Componente de Prueba
- Procedimiento de Prueba
- Plan de Prueba
- Defecto de Prueba
- Evaluación de la Prueba.

En RUP se definen diferentes niveles de prueba, yendo desde la menor unidad del programa con la realización de pruebas unitarias, se realizan pruebas de integración luego de integrar cada unidad probada, después se realiza la ejecución de pruebas de sistema (dentro de la cual se incluyen las pruebas de funcionalidad, stress, usabilidad, seguridad, configuración, instalación, recuperación, documentación, entre otras) y por último se llevan a cabo las pruebas de aceptación, en las que participan los usuarios finales.

Las pruebas están presentes en todo el ciclo de vida de RUP. Durante la fase de Inicio, se pueden planificar inicialmente las pruebas a partir de la Especificación de Casos de Uso del Sistema.

En la fase de Elaboración, el análisis y diseño del software posibilitan la determinación, de forma más específica, de las pruebas a realizar. En esta fase, una vez que se hayan implementado las primeras funcionalidades, se les realizan pruebas unitarias y funcionales.

En la fase de Construcción, se concluye la implementación de todas las funcionalidades, obteniéndose un producto el cual es sometido a pruebas de integración y de sistema.

En la fase de Transición se realizan las pruebas de aceptación en las que juegan un papel decisivo los usuarios finales. En este momento es donde se decide si se acepta o no la solución de software propuesta, es decir, si responde a los requisitos establecidos por el cliente.

El concepto de hito en RUP es un concepto que provee de elementos tangibles (artefactos), que permiten decidir si el trabajo realizado hasta el momento es el adecuado o no, y si se han superado las necesidades de cada fase definida en la metodología.

RUP proporciona una guía para ordenar las actividades de un equipo, dirige las tareas de cada desarrollador por separado y del equipo como un todo, especifica los artefactos que deben desarrollarse y ofrece criterios para el control y la medición de los productos y actividades del proyecto. Aunque corresponde a una metodología de trabajo intensivo en recursos, su aproximación al problema no sólo garantiza que los proyectos abordados serán ejecutados íntegramente sino que además evita desviaciones importantes con respecto a los plazos y, tan importante como esto, permite una definición acertada del sistema en un inicio para hacer innecesarias las reconstrucciones parciales posteriores.

1.7.2. Metodología Ágil: XP

XP es una metodología ágil basada en cuatro principios: simplicidad, comunicación, retroalimentación y valor. Además está orientada por pruebas y refactorización ya que estas se diseñan e implementan antes de programar la funcionalidad, creando el programador sus propias pruebas de unidad. Esta metodología es utilizada en proyectos cortos e invierte la mayor parte del tiempo en el desarrollo, probando y rectificando hasta conseguir el resultado adecuado.

Para XP, las características del software que no pueden ser demostradas mediante pruebas, simplemente no existen. Las pruebas dan la oportunidad de saber si lo implementado es lo que en realidad se tenía en mente.

En la metodología XP, el desarrollo de las pruebas es puesto en marcha por tres roles fundamentales:

- Los programadores: diseñan, programan y realizan las pruebas unitarias.
- El cliente: escribe los tests funcionales para determinar cuándo está completo un determinado aspecto.
- El probador: ayuda al cliente con las pruebas funcionales, asegurándose de que estas se ejecutan.

Los casos de prueba se escriben antes que el código. Las iteraciones son relativamente cortas ya que se piensa que entre más rápido se le entreguen desarrollos al cliente, más retroalimentación

se va a obtener y esto va a representar una mejor calidad del producto a largo plazo. Cada iteración incluye diseño, codificación y pruebas, fases superpuestas de tal manera que no se separen en el tiempo.

Las metodologías ágiles como XP plantean aumentar la velocidad del proyecto. Reducen la documentación (esto puede ser riesgoso con proyectos de complejidad alta), ya que para estas metodologías lo más importante es el desarrollo de la aplicación.

Estas metodologías han emergido como una interesante alternativa para proyectos pequeños, principalmente por los pocos artefactos que deben ser generados y al estilo de trabajo menos protocolar. Muchas personas con experiencia en metodologías tradicionales, suelen poner en duda la tenacidad y disciplina de una metodología ágil, ya que piensan que lo único que se hace en estas metodologías es programar.

1.8. Estado actual de las pruebas de software

Un pilar fundamental para garantizar la calidad lo constituyen las pruebas de software. La mayoría de las organizaciones que se dedican al desarrollo de software, ven las pruebas como un proceso opcional, sin importancia, lo que es un gran error para los que aún piensan así.

Actualmente se ha comprobado la utilidad de las pruebas de software. Sin embargo, la correcta implementación de los procesos o equipos de pruebas puede causar la falsa ilusión de que de estos depende la calidad del producto y la efectividad del proceso. Las pruebas de software tienen similar importancia a las demás actividades que intervienen en las etapas de desarrollo; pero su eficacia y eficiencia dependerá del conocimiento del equipo de trabajo, del modelo implantado, de la disciplina y de la organización para seguir el modelo, de los recursos asignados, de las habilidades del equipo de trabajo, entre otros. Esto implica que la implementación del proceso de pruebas debe ser considerada como un proyecto a largo plazo, no debe en ningún momento ser tratado como un proceso trivial o "de moda".

Hoy en día han surgido tendencias en virtud de demostrar la importancia de las pruebas de software, así como proponer mejoras a las mismas y de esta forma, incrementar la calidad del proceso de desarrollo. Entre estas tendencias se encuentran vincular el proceso de pruebas a lo largo del ciclo de vida de desarrollo de software y la automatización de las mismas.

1.8.1. Automatización de las pruebas de software

Para la automatización de las pruebas se han creado una serie de herramientas que ayudan en gran medida a su realización, ya que en nuestros días los sistemas que se necesitan son cada vez más grandes y complejos y con estas herramientas se puede reforzar el desarrollo de todo el proceso de pruebas, disminuyendo tiempo, esfuerzo y costo.

Entre las herramientas creadas para analizar el código se encuentra JUnit (para aplicaciones en Java). Esta ha tenido tanto éxito que se ha extendido a otros muchos lenguajes de programación. Todos los frameworks heredados de JUnit han recibido la denominación xUnit.

Entre los framworks xUnit, existen versiones para C/C++ (CUnit y CPPUnit), Delphi (DUnit), PHP (PHPUnit), HTML (HTMLUnit), NUnit (plataforma .NET), VBUnit (Visual Basic) entre otros.

Todas estas herramientas ayudan al diseño y ejecución de las pruebas de Caja Blanca. Estas constituyen un conjunto de framework que permiten realizar la ejecución de clases de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera.

NUnit, como caso particular, es un framework desarrollado en C# que ofrece las funcionalidades necesarias para implementar pruebas en un proyecto. Provee una interfaz gráfica para ejecutar y administrar las mismas, informando si una prueba (o un conjunto de pruebas) falló, pasó o fue ignorada. Para ello compara los valores esperados y los valores generados, y si estos son diferentes la prueba no pasa, en caso contrario la prueba es exitosa.

1.8.2. Metodologías, modelos y estándares

La calidad del software, más específicamente el proceso de pruebas, ha tenido un gran auge a nivel mundial y constituye actualmente un especial interés para la comunidad de Ingeniería de Software. Con el objetivo de vincular el proceso de pruebas a lo largo del ciclo de vida de desarrollo de software y mejorar la calidad, se han creado un conjunto de metodologías, modelos y estándares. Esto constituye también una alternativa para alcanzar competitividad en esta industria.

Entre los modelos y estándares desarrollados se encuentran CMM, CMMI, ISO/IEC y SPICE, además de la norma ISO 9001:2000, que es un sistema para la gestión de la calidad. Existen diferentes opiniones acerca de que estos programas de mejora de procesos sólo son posibles para empresas grandes que son las que cuentan con los recursos suficientes para enmarcarse en este tipo de prácticas, ya que los estándares de mejora propuestos internacionalmente por

organismos como el SEI (Software Engineering Institute) e ISO (International Organization for Standardization) no han sido creados para las pequeñas y medianas empresas sino para empresas grandes.

GreenSQA (empresa de ingeniería que brinda servicios de aseguramiento de calidad de pruebas de software) define una metodología de acuerdo al estándar IEEE 1012-1986 que indica el conjunto mínimo de pruebas que se deben realizar a un sistema. Este propone las siguientes: prueba modular, prueba unitaria o prueba de componentes, prueba de integración, prueba del sistema y prueba de aceptación.

Sopra PROFit[2] (16) estima que el 75% de las organizaciones dedicadas a la producción de software tienen un nivel de madurez bajo en la realización de procesos de pruebas y que a medida que las organizaciones van alcanzando mayores niveles de madurez, mayor es la importancia que dan a la fase de pruebas en el desarrollo de software. Como respuesta a esta problemática, PROFit ha desarrollado un modelo de Testing basado en el modelo TMap (Testing Management Approach) que estructura el proceso de prueba, abarcando el ciclo completo del mismo y que además cumple con los requisitos especificados en el modelo CMMI[3], uno de los más extendidos en las áreas de desarrollo.

1.8.3. Calidad de software en Cuba

La calidad del software cubano y el apoyo gubernamental al desarrollo digital hacen de nuestro país un importante socio comercial (17) Según opiniones de algunos interesados en el tema, los productos cubanos cuentan con un gran prestigio a nivel internacional, por la adecuada elección de los ambientes informáticos (plataformas) en los cuales se sustentan los proyectos, así como la compatibilidad o adaptabilidad a las diferentes líneas desarrolladas en el mercado mundial.

Entre las empresas cubanas que se desarrollan en esta área se encuentran las empresas Segurmática y Softel, la primera dedicada a la creación de antivirus para proteger los sistemas informáticos cubanos, y la segunda dedicada a la producción y comercialización de software, desarrollando sistemas informáticos en una amplia gama de aplicaciones como el turismo y la medicina.

[2] Programa de Fomento de la Investigación Técnica.

[3] Capability Maturity Model Integration (Modelo de Integración de la Capacidad de Madurez del proceso de desarrollo de software).

Desoft es otra de las empresas cubanas dedicada a prestar servicios de desarrollo, producción y comercialización de software. Esta empresa tiene una oficina central y una sede en cada una de las provincias del país. Una de sus tareas relacionadas con la calidad es la creación de normas para asegurar la eficiencia de los productos.

En nuestro país también han tenido lugar distintos talleres sobre la calidad de las Tecnologías de la Información y las Comunicaciones (TIC), en los que se han debatido temas como la importancia de las pruebas de software y las distintas tendencias que existen en la actualidad sobre este aspecto. Aproximadamente cada dos años se efectúa en Cuba el evento Informática. Su primera edición fue en 1988 donde participaron 180 delegados extranjeros y alrededor de 878 delegados cubanos, y su última edición fue en el año 2005. Este evento ha tenido participación permanente de la UNESCO, de numerosas instituciones, universidades y centros de investigación de Iberoamérica. Con estos eventos se puede afirmar que Cuba ha vencido el reto de aplicar tecnologías informáticas con pocos recursos económicos.

Dentro de esta esfera de la producción de software se enmarca nuestra Universidad, la cual no sólo ha tenido como prioridad la informatización de nuestro país sino también la exportación de productos. Algunos países como Venezuela han sido beneficiados ya con los logros obtenidos.

Para el aseguramiento de la calidad en la Universidad se creó la Dirección de Calidad del Software, llevando esta el control de todos los proyectos que se han ido desarrollando. Con este propósito este grupo creó el expediente del proyecto, elaborado con guías, estándares y normas internacionales como ISO e IEEE, CMMI y la documentación de RUP. Entre sus objetivos fundamentales se encuentra lograr homogeneidad en los proyectos productivos de la universidad. Con el propósito de mejorar el proceso de prueba, como un elemento importante de la calidad, se estableció también un laboratorio de pruebas, creando de esta forma un entorno adecuado para esta actividad.

Para la gestión de la calidad del software a nivel de proyecto, la Dirección de Calidad de la Infraestructura Productiva de la Universidad, propuso un conjunto de lineamientos a seguir. Estos pueden ser aplicables a cualquier proyecto, ajustándose a cada caso particular. En estos lineamientos se le dedica un espacio a las pruebas de software, quedando recogida la necesidad de definir roles que generalmente se obvian, como es el caso del ingeniero de componentes, diseñador de pruebas y probador. También se propone en la etapa de implementación del sistema, la realización de actividades tales como: Implementación de elementos de prueba y Ejecución de pruebas de unidad, y la documentación de artefactos como Componentes de prueba.

Aunque en la Universidad no se tenía experiencia en el proceso de calidad del software, en estos últimos tiempos se ha visto un gran interés por parte del grupo central de calidad, que ha hecho un gran esfuerzo por motivar a los proyectos productivos a llevar a cabo de la mejor manera posible el proceso de pruebas, y demás actividades que conlleva la obtención de un mejor producto.

1.9. Conclusiones

Tras el estudio realizado en este capítulo se ha arribado a las siguientes conclusiones:

- Actualmente el número de empresas y organizaciones que se dedican a la producción de software ha ido aumentando significativamente. De esta forma se les han ido dando a las pruebas de software la importancia y significación que tienen estas, como requisito básico para la mejora continua de la calidad.
- Cuba, al igual que el resto de los países que se desarrollan en esta área, ha mostrado un gran interés por las pruebas del software como parte fundamental de la calidad. Muestra de ello son las empresas cubanas que han sido creadas con este propósito.
- Durante los últimos años se han creado un conjunto de herramientas para la automatización de las pruebas. Estas permiten, junto con las metodologías de software y las técnicas existentes, llevar a cabo un proceso de pruebas con mayor calidad.

Capítulo 2: Características de la estrategia

2.1. Introducción

Este capítulo se centra en la selección de los procesos que formarán parte de la estrategia. Para ello se definen primeramente los niveles de prueba a realizar, resaltando la significación que tienen en estos las pruebas de Caja Blanca y Caja Negra. Como parte de la selección de la metodología de desarrollo a utilizar, se hace un breve análisis de las ventajas que proporciona la metodología RUP y de su correcto empleo en el proyecto.

Con motivo de obtener un mayor conocimiento acerca de los principales roles que deben participar en la etapa de pruebas y de los artefactos que se deben generar como salida de cada actividad, en este capítulo se realiza un análisis de los mismos, destacando el papel que desempeñan y su importancia. Como parte fundamental de la estrategia se seleccionan los procesos, así como sus entradas, actividades, salidas y roles que intervienen en cada uno de ellos. Por último se hace una proposición de las herramientas a utilizar para la automatización de las pruebas y de las métricas más importantes para medir la completitud del proceso.

2.2. Definición de la estrategia

Algunos autores como Krutchen (18), Pressman (6), Pfleger (19), Cardoso (20) y Sommerville (21) afirman que el proceso de ejecución de Pruebas debe ser considerado durante todo el ciclo de vida de un proyecto, para así obtener un producto de alta calidad. Su éxito dependerá del seguimiento de una estrategia de prueba adecuada. La estrategia de prueba de software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba: la planificación, el diseño de casos de prueba, la ejecución y los resultados, tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software.

Para realizar esta estrategia se debe partir de un diagnóstico en el que se evidencie el problema. Luego se realizará la proyección y ejecución de acciones progresivas que permitan alcanzar paulatinamente los objetivos propuestos.

Con el objetivo de obtener cual es la situación actual de las pruebas en el proyecto Registros y Notarías, se le realizó una entrevista a la líder del proyecto. De esta encuesta se obtuvo lo siguiente:

- Actualmente el proyecto Registros y Notarías cuenta con un equipo de seis personas encargados de realizar las tareas vinculadas a la calidad interna. Estos son los que diseñan las pruebas, las aplican y obtienen el documento con los errores encontrados, sin tener correctamente un rol específico definido. El responsable del equipo redacta luego el Documento Único de No Conformidades que es entregado a los desarrolladores, para su posterior corrección.
- Para la automatización de las pruebas no existen herramientas, por ejemplo para la automatización de las pruebas de Caja Blanca.
- En el proyecto no se realizan las pruebas de Caja Blanca como es debido. Los desarrolladores revisan el código escrito sin un procedimiento adecuado, restándole la importancia que requiere esta tarea.

Para lograr un mejor desarrollo y aplicación de las pruebas en el proyecto Registros y Notarías, se definirán los pasos a seguir para realizar de una mejor manera las pruebas de Caja Blanca y Caja Negra, logrando con ello la obtención de un producto con mayor calidad.

Para ello es necesario conocer que: toda estrategia de prueba de software debe incluir pruebas de bajo nivel que verifiquen que todos los pequeños segmentos de código se han implementado correctamente, así como las pruebas de alto nivel que validen las principales funciones del sistema frente a los requisitos del cliente. La misma debe proporcionar una guía al profesional y un conjunto de hitos para el jefe del proyecto.

2.2.1. Alcance

Además de la existencia de otros tipos de pruebas, la estrategia propuesta se basa en la realización de las pruebas de Caja Blanca y Caja Negra al proyecto Registros y Notarías. Para el desarrollo de la misma se analizan los diferentes procesos que intervienen en la etapa de pruebas, así como los artefactos que se deben generar, los roles que deben intervenir en cada uno de los procesos, entradas, actividades y salidas del mismo. Mediante una representación gráfica se explica con más detalle el orden en que deben realizarse cada una de las actividades.

En la estrategia no se define el momento en que deben culminar las pruebas, simplemente se realiza una propuesta para cada iteración. No obstante, se propone al Responsable de calidad interna y al Responsable del equipo de desarrollo que se le realice al producto la cantidad de iteraciones necesarias mientras se encuentren defectos críticos en la aplicación. (Ver Anexo 6)

2.3. Niveles de prueba propuestos

Para llevar a cabo la estrategia con éxito, se debe comenzar a trabajar desde el interior del programa, es decir, desde la partícula más pequeña hasta la máxima expresión que el mismo pueda alcanzar; por lo que se decide comenzar este proceso realizando la prueba de unidad. El énfasis de esta prueba es verificar que esta pequeña unidad funcione correctamente en forma aislada antes de proceder a integrarla al sistema. La prueba de unidad esta basada fundamentalmente en la realización de técnicas de pruebas de Caja Blanca y en menor medida en técnicas de pruebas de Caja Negra. Para optimizar tiempo y esfuerzo se automatizan las técnicas de pruebas de Caja Blanca empleando para ello la herramienta NUnit.

Luego se pasaría a verificar que todas estas unidades que ya han sido probadas y funcionan correctamente, lo hacen de igual forma cuando interactúan y se integran con otras unidades del sistema. Después de haber verificado cada una de ellas individualmente, se espera que al interactuar estas, se mantengan funcionando sin problemas, lo que lamentablemente no ocurre siempre. En ocasiones, los datos se pueden perder en una interfaz, un módulo puede tener un efecto desfavorable e inesperado sobre otro, si se relacionan varias funciones de diferentes interfaces estas pueden no producir el resultado esperado, todos estos errores pueden presentarse al llevarse a cabo la integración.

Después que el software ha sido integrado se realizan pruebas de alto nivel, entre estas se encuentra la prueba de validación, que es de suma importancia ya que es la que proporciona la información de si se satisfacen o no todos los requisitos funcionales. Para la realización de esta prueba se usa exclusivamente la técnica de prueba de Caja Negra.

La realización de las pruebas de sistema está comprendida entre las pruebas de alto nivel. Este tipo de pruebas permiten probar el sistema como un todo así como también aspectos relacionados con la integración del producto a otros sistemas. En esta etapa se le pueden realizar también pruebas de Caja Negra al producto, aunque se le da mayor atención a las pruebas de estrés, rendimiento, configuración, seguridad, usabilidad, entre otras.

2.4. Metodología a utilizar

Actualmente en el proyecto Registros y Notarías se emplea RUP como metodología para el desarrollo del software orientado a objeto. Luego de haber realizado un estudio en el epígrafe 1.6 de las características más importantes de las metodologías RUP y XP y el tratamiento que le da

cada una de ellas a las pruebas, se considera que la utilizada en el proyecto es la más conveniente.

La utilización de RUP es factible en proyectos de gran tamaño ya que tiene un proceso de pruebas bien definido. La ventaja de ser dirigido por casos de uso posibilita el diseño de casos de prueba y la posible automatización de las mismas.

Debido a que es iterativo e incremental, las pruebas no se realizan cuando el producto está terminado, sino que este posibilita que en cada iteración pueda ser aplicado al menos un ciclo de pruebas.

El uso intensivo de la documentación es una buena práctica que no debe abandonarse.

Es por esto que RUP constituye una buena alternativa para los proyectos grandes, debido a los artefactos que brinda para la documentación.

2.4.1. Roles

Durante la etapa de pruebas deben intervenir cuatro roles fundamentales, los cuales son responsables de diseñar, aplicar y dar seguimiento a todas las pruebas a realizar. Estos se nombran a continuación:

Administrador de Pruebas: Es el máximo responsable del éxito de las pruebas, pues es el encargado de verificar y asegurar que se realice una correcta planificación y administración de los recursos. Comprueba el progreso y efectividad de las pruebas, evalúa los resultados de cada ciclo, y debe dar solución a los problemas que impiden el buen desarrollo del proceso. Es por ello que este debe ser seleccionado por sus aptitudes y capacidad para realizar dicha tarea, ya que será el que estará dirigiendo tan importante proceso.

Analista de Pruebas: Es responsable de identificar y definir las pruebas requeridas, supervisando el progreso de las pruebas, así como el resultado por cada ciclo. Tiene como responsabilidad la obtención de los datos de pruebas necesarios para obtener todas las posibles respuestas del sistema. Es encargado de los casos de prueba y de los procedimientos de prueba.

Diseñador de Pruebas: Es responsable de definir la estrategia que guiará el proceso para así asegurar su aplicación exitosa. El rol involucra identificar las técnicas apropiadas, herramientas y pautas para llevar a cabo las pruebas requeridas, y define los recursos necesarios. Este rol también es nombrado como Arquitecto de prueba o Arquitecto de automatización de prueba.

Probador: Este rol es responsable de las principales actividades y el mayor esfuerzo de las pruebas. Es quien ejecuta las pruebas y obtiene directamente los resultados. Se encarga de hacer un Registro donde se plasman todos los defectos encontrados como resultados de la ejecución de las pruebas. Participa en cada una de las iteraciones realizadas durante el proceso de desarrollo, comprobando primeramente el funcionamiento de los módulos y luego la integración de estos como sistema, validando finalmente que el producto obtenido tenga calidad.

Los roles mencionados anteriormente son propuestos por RUP a los cuales se suman los que se nombran a continuación, ya que las condiciones específicas del proyecto lo requieren para una mayor organización del trabajo.

Se definen dos roles, el Responsable de Calidad Interna y los Desarrolladores del Software, cada uno con su descripción y conocimientos necesarios para desempeñar las tareas que les sean asignadas.

Responsable de Calidad Interna: Es el encargado de planificar y organizar las revisiones, orientando los artefactos a revisar así como su entrega, también planifica los cursos de capacitación para el equipo y la aceptación del plan de pruebas. Se encarga de conformar el informe único de No Conformidades con todos los defectos encontrados y se lo hace llegar a los responsables de darle solución a los problemas. Este también debe realizar un resumen de los resultados a través de métricas archivándose los resultados para su posterior uso, finalizando así las revisiones.

Desarrollador del software: Participa junto al Analista de Pruebas en el diseño de los casos de prueba para las pruebas de unidad y de integración. También participa en la implementación y ejecución de las pruebas empleando la herramienta NUnit junto al probador y es el encargado de realizar la depuración.

2.4.1.1. Perfil de competencia

Para el buen desempeño del personal que ocupará los roles que se proponen para el desarrollo de la estrategia se necesita que estos tengan algunos conocimientos básicos los cuales se relacionarán a continuación.

Rol	***Conocimientos Mínimos***
Administrador de Pruebas	Metodología RUP

	Pruebas de Software Ingeniería de Software Conocimientos Básicos sobre el negocio Métricas de prueba
Analista de Pruebas	Metodología RUP UML Ingeniería de Software Pruebas de Software
Diseñador de Pruebas	Metodología RUP UML Ingeniería de Software Pruebas de Software
Probador	Conocimientos sobre el negocio. Conocimientos básicos de programación. Metodología RUP Ingeniería de Software
Responsable de calidad interna	Metodología RUP Pruebas de Software Ingeniería de Software Conocimientos Básicos sobre el negocio Métricas de prueba
Desarrollador	Pruebas de Software Automatización de las pruebas

2.4.2. Artefactos

Los artefactos son productos tangibles del proyecto, que son producidos, modificados y usados por las actividades (6).

Según la metodología RUP, cada uno de los procesos de la etapa de pruebas debe generar un conjunto de artefactos. Para llevar a cabo la estrategia se definieron los que se consideran más importantes y necesarios con el objetivo de efectuar un buen proceso de prueba. Los que formarán parte en esta estrategia son los siguientes:

Plan de prueba

El plan de pruebas dirige, orienta y restringe las operaciones a realizar durante la etapa de pruebas, centrando el trabajo en entregables útiles y necesarios. En este artefacto se incluye el propósito y alcance, qué requerimientos se van a probar, las herramientas a utilizar, los recursos a emplear, reflejando las características de hardware y software, el cronograma, así como el documento que va a ser entregado.

Deben quedar definidos en este artefacto:

- Propósito
- Alcance
- Referencias
- Requerimientos a probar
- Herramientas
- Recursos

Estrategia de prueba

El artefacto estrategia de prueba describe los objetivos generales de las pruebas. Incluye las fases de prueba que se deben seguir y los tipos de pruebas que se deben realizar.

Deben quedar definidos en este artefacto:

- Propósito
- Alcance
- Entregables
- Técnicas de prueba

Configuración del entorno de prueba.

En el artefacto Configuración del entorno de prueba se detalla la distribución de hardware y software para llevar a cabo las pruebas, periféricos de entrada y salida y sus drivers correspondientes, así como herramientas que faciliten la prueba.

El objetivo de este artefacto es separar el ambiente de pruebas del ambiente de desarrollo e instalar las herramientas necesarias y el software a probar en la versión correspondiente a cada ciclo de vida de prueba.

Deben quedar definidos:

- Propósito
- Alcance
- Inventario de recursos

Datos de pruebas

Para diseñar y aplicar los casos de pruebas se necesitan datos (válidos e inválidos) que ayuden a la ejecución de los mismos, permitiendo de esta forma que el sistema se ejecute en todas sus variantes. Los datos de prueba no son más que datos reales referentes al negocio correspondiente, los cuales se utilizan como entrada, salida o precondiciones. Estos se escogen atendiendo a las especificaciones del negocio. Un caso de prueba debe ejecutarse una vez por cada combinación de estos valores.

Ejemplo de datos de prueba:

Cadenas

- cadena vacía
- cadena con un espacio vacío
- valores cortos y largos
- valores válidos e inválidos
- caracteres o combinaciones ilegales
- caracteres especiales como #, ", ', &, @, etc.

Números

- cadena vacía, si es posible
- 0
- pequeños y largos en rangos positivos
- pequeños y largos en rangos negativos

- fuera del rango de positivos
- fuera del rango de negativos
- que comiencen con ceros
- letras

Los datos de prueba deben contener:

- Campos de la interfaz
- Datos válidos
- Datos inválidos

Casos de pruebas

Los casos de prueba constituyen la especificación formal donde quedan registrados los datos de entrada de la prueba, las condiciones para su ejecución, así como los resultados previstos.

Un caso de prueba incluye la verificación del resultado de la interacción entre los actores y el sistema, las precondiciones y poscondiciones especificadas por el caso de uso, así como la secuencia de acciones especificadas por este.

Para el buen diseño de un caso de prueba, deben incluirse los siguientes aspectos:

- Fecha en la que se diseña o aplica el caso de prueba.
- Versión del producto que se está probando.
- Descripción de lo que se va a realizar (diseñar o aplicar el caso de prueba).
- Autor(es) que participa(n) en la realización del caso de prueba.
- Introducción.
- Propósito del caso de prueba.
- Alcance que tendrá el caso de prueba.
- Una sección para exponer las Definiciones, Acrónimos y Abreviaturas que se utilizarán en el caso de prueba.
- Una sección para las Referencias, que son los documentos que servirán de apoyo a la realización de los casos de prueba.
- Pruebas Realizadas al Caso de Uso: En esta sección se recogen los casos de prueba que fueron desglosados por escenarios para ese caso de uso.

Cada caso de prueba obtenido por los distintos escenarios del caso de uso debe llevar implícito:

- El nombre del caso de prueba.
- La descripción de la funcionalidad que se va a probar.
- El flujo central que muestra la interacción entre el actor y el sistema.
- Las condiciones que son necesarias para la ejecución de la prueba.
- Una sección para las iteraciones, en la que se deben mostrar las clases válidas e inválidas, el resultado que se espera y el resultado final de la prueba (satisfactorio o insatisfactorio).
- Una sección para el Registro de defectos y dificultades detectados (no conformidades).
- Y por último una sección para los anexos donde se recogen las imágenes de las no conformidades.

Procedimientos de prueba

El artefacto procedimiento de prueba constituye el conjunto de instrucciones detalladas para la disposición y ejecución paso a paso de uno o más casos de prueba. Este puede constituir una instrucción sobre como ha de realizarse un caso de prueba manualmente o puede ser una especificación de cómo interactuar con una herramienta de automatización de pruebas.

Cada caso de prueba llevará asociado un procedimiento de prueba con las instrucciones para realizar la prueba.

En el caso de las pruebas de Caja Blanca, cada caso de prueba tendría asociado un procedimiento de prueba con los pasos a seguir para utilizar la herramienta NUnit, herramienta que se propone para su automatización.

Sumario de evaluación de las pruebas

En este artefacto se expone un resumen de los resultados de las pruebas aplicadas a los componentes operacionales del sistema, lo cual permite revisar y evaluar la calidad del producto que se está desarrollando. Este artefacto puede contener observaciones y recomendaciones para futuros ciclos de pruebas, a lo que se suman los siguientes aspectos:

- Breve descripción del contenido del Sumario de Evaluación.
- Resumen de los resultados de las pruebas.
- Análisis de la cobertura de las pruebas.

2.5. Marco de procesos

Para la elaboración de la estrategia se analizaron los procesos asociados a las pruebas de software. De esta forma se propone la realización de 5 procesos, como muestra la figura 10, que se consideran necesarios e indispensables para el desarrollo y progreso efectivo de cada una de las acciones a llevar a cabo en la etapa de pruebas. Los procesos son los siguientes:

1- Planificación de las pruebas

2- Diseño de las pruebas

3- Implementación de las pruebas

4- Ejecución de las pruebas

5- Evaluación de las pruebas

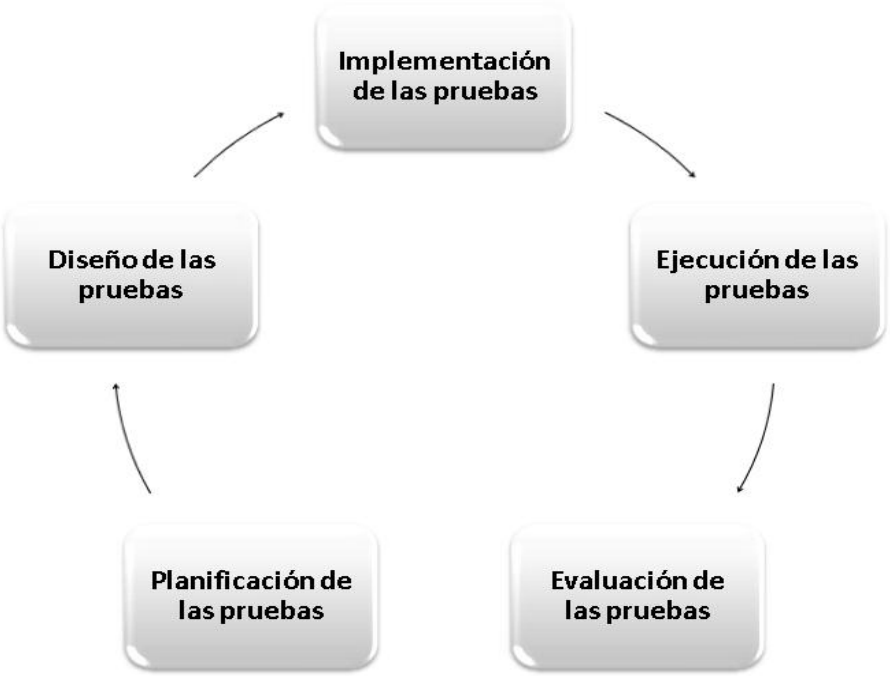


Figura 9: Procesos de la estrategia

Una fase está compuesta por un ciclo de varias iteraciones. Por tanto, es necesario comprender que para cada nueva integración de funcionalidades que se le vaya realizando al sistema en desarrollo, donde se van incorporando sucesivamente los casos de uso, se realizarán nuevos diseños de casos de pruebas y se obtendrán nuevos resultados. Es por esto que algunos de estos procesos, mencionados anteriormente, describen un ciclo de desarrollo para las pruebas.

Para documentar los procesos el formato propuesto incluye: Nombre, Gráficos, Objetivo, Roles que participan, Definiciones, Entradas, Actividades, Salidas y Recursos.

2.5.1. Planificación de las pruebas

En la etapa de planificación se definen las pruebas que se les realizarán a la aplicación y se identifican los participantes y sus roles, designando un responsable. En este proceso se debe obtener el esfuerzo y tiempo necesario para las pruebas, productos entregables, recursos requeridos, repositorios y ambientes de prueba, entre otros aspectos.

El plan de pruebas debe especificar también hitos de control y de seguimiento, en los cuales el responsable de las pruebas debe realizar un análisis de lo sucedido en el proceso y mostrar los resultados al grupo del proyecto y a la dirección del mismo. Del control y medición que se realice, depende en gran medida el progreso del producto y del proceso de desarrollo.

En la realización del plan de pruebas se describe de forma concisa y clara como el equipo de calidad interna realizará todas sus pruebas. Cómo se desarrollará su flujo de trabajo, es decir, su planificación, roles que intervendrán en el proceso de pruebas y funcionamiento de los mismos y herramientas que serán empleadas, todos estos elementos deben quedar bien definidos.

Es confeccionado teniendo como base lo que propone RUP, sólo que ha sido adaptado a las condiciones específicas del proyecto de forma general. Por tanto, su uso es muy particular del proyecto.

Uno de los principios más importantes de las pruebas de software es que estas deben planificarse mucho antes de que comiencen a desarrollarse.

Por esta razón el plan de pruebas debe iniciarse en la fase de Elaboración, pues aquí se planifican las pruebas que se necesitan en cada iteración durante el desarrollo del sistema.

En esta fase ya se debe obtener un producto con un determinado nivel. De esta forma, a medida que se vayan implementando algunas funcionalidades, se hará necesario comenzar a planificar las pruebas, pues entre más rápido se le entreguen los errores encontrados a los desarrolladores, más rápido se le dará solución a estos, lo que representa una mejor calidad del producto a largo plazo.

Objetivo

El objetivo principal de la planificación de las pruebas es definir los roles y recursos del sistema para un apropiado entorno de pruebas, así como la definición de una estrategia adecuada y la planificación del esfuerzo.

Roles que participan

- Administrador de prueba.
- Diseñador de prueba.
- Responsable de calidad interna

Entradas

- Requerimientos.
- Modelo de casos de uso.
- Modelo de diseño.
- Modelo de implementación.

Actividades

- Identificar objetivos de las pruebas.
- Evaluar recursos necesarios.
- Definir los tipos de prueba a realizar.
- Definir las pautas de las pruebas.
- Definir la configuración del entorno de prueba.

Salidas

- Artefacto Plan de Prueba.
- Artefacto Configuración del Entorno de Prueba.
- Artefacto Estrategia de Prueba.

Recursos

Para realizar el plan de pruebas, en el proyecto Registros y Notarías se emplea la plantilla propuesta por la Dirección de Calidad de la Universidad. A la misma se le han realizado los ajusten necesarios para que esta vaya en correspondencia con las características del proyecto.

Esta plantilla cuenta con los siguientes aspectos:

- Revisiones históricas
- Tabla de contenidos

- Introducción
    - Propósito
    - Alcance
    - Definiciones, Acrónimos y Abreviaturas
    - Referencias
    - Resumen
- Organización
- Flujo de trabajo
- Metodología de aplicación
- Niveles de prueba
- Anexos

Representación Gráfica

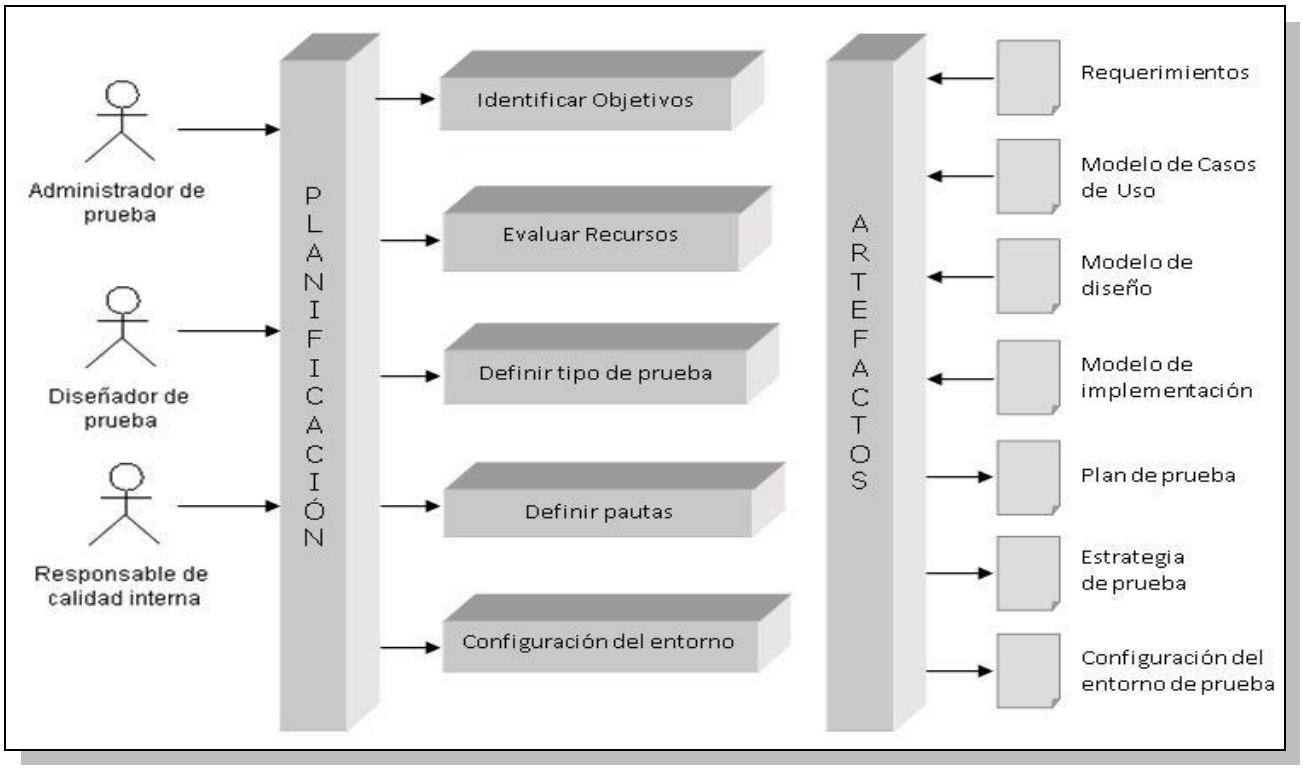


Figura 10: Planificación de las pruebas

2.5.2. Diseño de las pruebas

Una vez que se ha realizado la planificación de las pruebas se pasaría a la etapa del diseño de las mismas, ya que el plan de pruebas es una de las entradas más importantes para este proceso.

A partir de la especificación del producto se diseñan los casos de prueba, y se identifican los posibles datos de prueba que serán utilizados posteriormente en la ejecución de las mismas. En esta etapa deben quedar estructurados los procedimientos de pruebas necesarios para lograr una ejecución satisfactoria.

Un buen plan de pruebas, para ser ejecutado correctamente debe contener casos de prueba que cumplan con las siguientes características. Deben ser precisos, pues las pruebas deben describir correctamente lo que verificarán; económicos, ya que sólo contendrán los pasos necesarios para su propósito; repetibles, pues deben ser consistentes y recoger todo lo necesario para que los resultados de cualquier ejecución sean los mismos; adecuados y trazables, ya que deben describir la situación en la que son aplicables y deben estar relacionados con los requisitos funcionales que cubren, con objeto de facilitar la identificación del fallo y permitir hacer un seguimiento de las correcciones más fácilmente.

El proceso de diseño de las pruebas se realiza cada vez que haya una nueva iteración, por esta razón será necesario diseñar casos de prueba en cada fase que lo necesite, ya sea en la fase de Elaboración, Construcción o en la fase de Transición.

Objetivo

La etapa de diseño de pruebas tiene como objetivos principales identificar los casos de prueba y estructurar los procedimientos de pruebas.

Roles que participan

- Analista de pruebas.
- Desarrollador del software.

Entradas

- Requerimientos.
- Modelo de casos de uso.
- Modelo de diseño.
- Modelo de implementación.
- Plan de Pruebas.

- Estrategia de Prueba.

Actividades

- Identificar los datos de prueba.
- Identificar y describir los casos de prueba.
- Identificar y estructurar los procedimientos de prueba.

Salidas

- Artefacto Casos de prueba.
- Artefacto Procedimientos de prueba.
- Artefacto Datos de prueba.

Recursos

En el proyecto Registros y Notarías se utilizaba anteriormente una plantilla de casos de prueba donde se mostraban las acciones por medio de clases válidas e inválidas, y finalmente se recogían los errores en una tabla denominada Registro de defectos y dificultades detectados.

Actualmente el grupo de calidad de la Universidad propuso una nueva plantilla para ser tomada como estándar en todos los proyectos. El empleo de esta constituye una ventaja pues además de que está mejor estructurada, en esta se especifican cada una de las variables probadas durante todo el proceso, y los diferentes datos de prueba empleados. Además en la misma se plasman mejor las no conformidades detectadas, y esto le sirve al desarrollador para corregir todos los defectos encontrados, facilitándole su localización.

Por estos beneficios se considera que para el diseño y aplicación de los casos de prueba en el proyecto se debe emplear esta plantilla que constará de los siguientes aspectos:

- Nombre del Proyecto
- Nombre del Módulo
- Versión del proyecto
- Nombre del caso de uso a probar
- Versión del caso de prueba
- Control de Versiones
- Tabla de Contenido

- Descripción General
- Condiciones de ejecución
- Secciones a probar en el caso de uso
- Sección a revisar
- Registro de defectos y dificultades detectados

Representación Gráfica

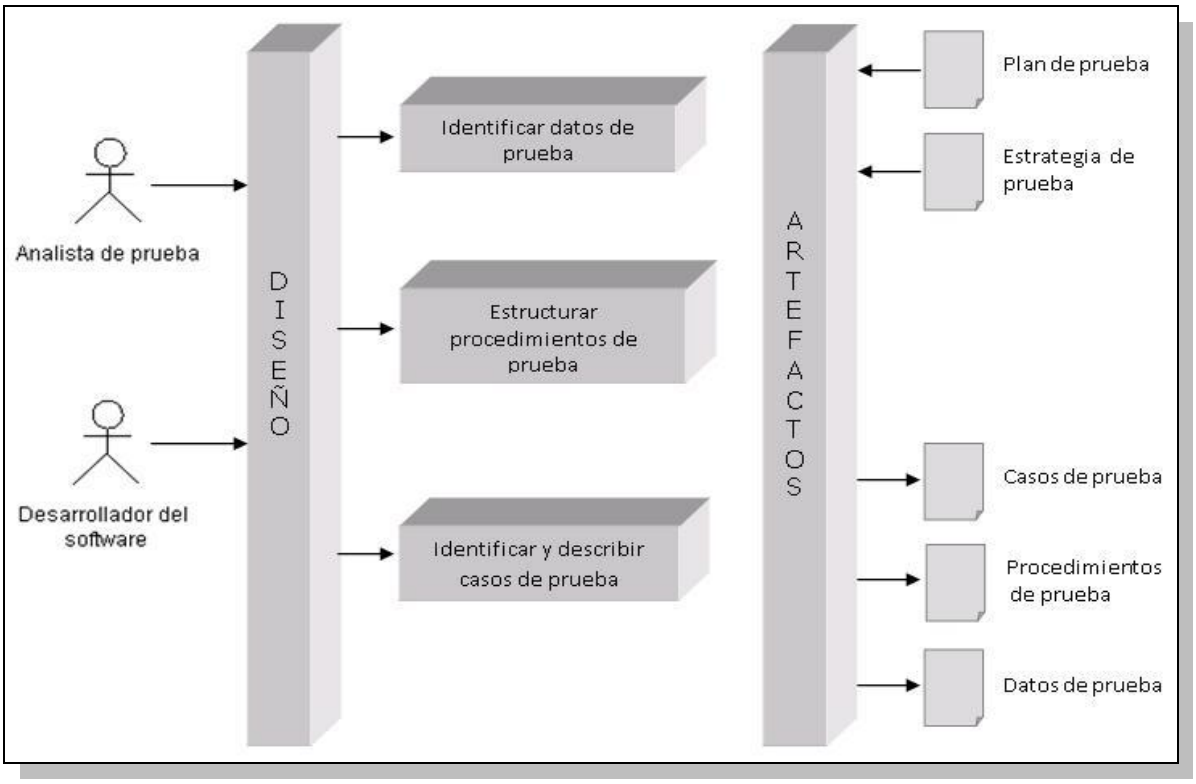


Figura 11: Diseño de las pruebas

2.5.3. Implementación de las pruebas

Siempre que una prueba pueda ser automatizada, se debe proceder a la selección de herramientas que posibiliten que el proceso de pruebas se desarrolle de forma rápida y eficiente. Estas herramientas no sólo asisten a las pruebas automatizadas, también brindan soporte para la organización de los artefactos del proyecto.

El proceso de implementación de las pruebas en un proyecto determinado puede o no realizarse, esto está en dependencia de que se cuente o no con herramientas adecuadas para la automatización y que el personal esté capacitado para utilizar dicha herramienta.

El empleo de estas herramientas en el proyecto Registros y Notarías es muy recomendable debido a que el sistema que se encuentra actualmente en desarrollo es muy grande, y con el uso de estas se puede reforzar el proceso de pruebas, disminuyendo tiempo, esfuerzo y costo.

Objetivo

El propósito de esta etapa es la automatización de los procedimientos de prueba obteniendo los componentes de prueba correspondientes.

Roles que participan

- Diseñador de pruebas.
- Probador.
- Desarrollador del software.

Entradas

- Caso de prueba.
- Procedimiento de prueba.
- Modelo de implementación.

Actividades

- Identificar mecanismos de prueba.
- Definir los elementos de prueba.
- Implementación de las pruebas.

Salidas

- Artefacto Componentes de prueba.

Recursos

Para la aplicación de las pruebas de Caja Blanca se empleará la herramienta NUnit para optimizar el proceso. De esta forma también se realiza un mayor número de pruebas lo cual es una gran ventaja que proporciona la misma.

Representación Gráfica

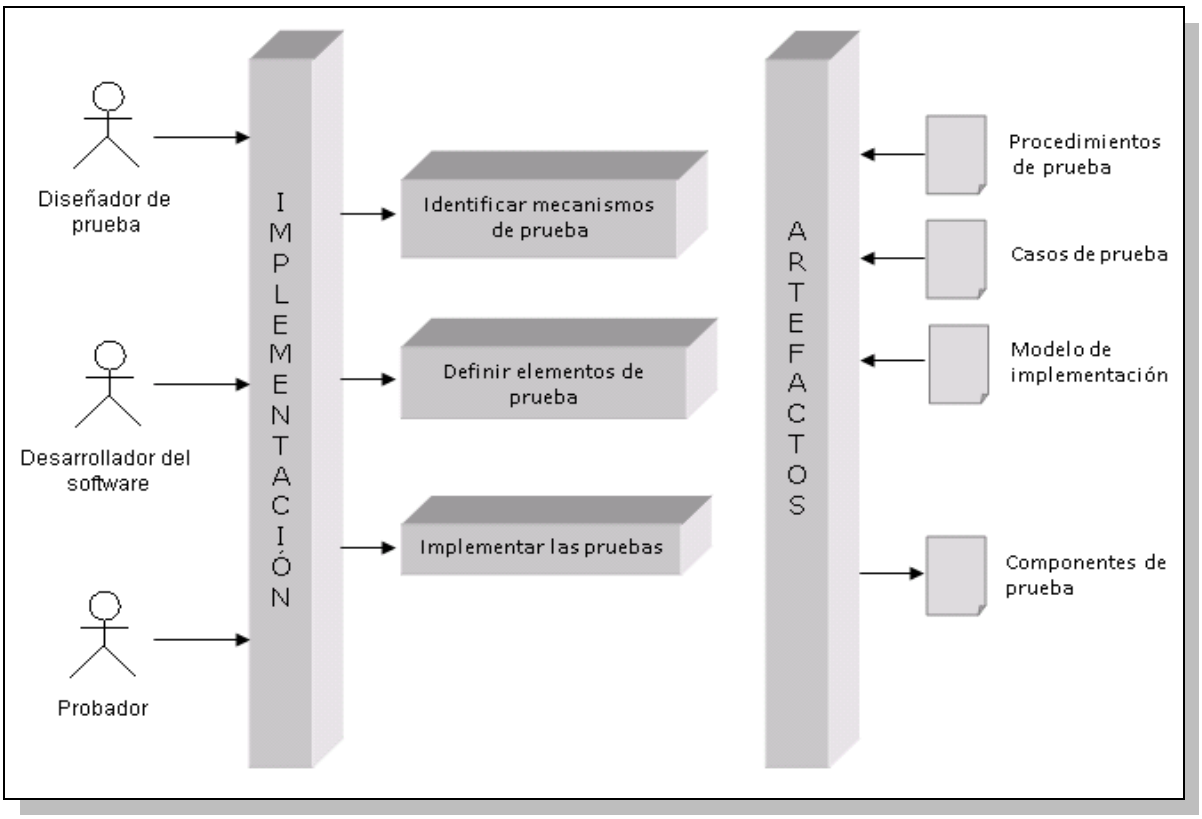


Figura 12: Implementación de las pruebas

2.5.4. Ejecución de las pruebas

Al tener bien definidos los casos de prueba, los datos de prueba y la estrategia a seguir, ya el probador puede ejecutar las mismas con el objetivo de detectar los errores existentes en el software.

Independientemente de la forma en que se realicen las pruebas, ya sean manuales y/o automáticas, y de los tipos de pruebas que se acordaron en el alcance del proyecto, la ejecución de pruebas se basa en los ciclos o iteraciones, en los cuales se ejecutan los conjuntos de casos de pruebas diseñados, en busca de una estabilidad incremental del producto.

Objetivo

El objetivo de esta etapa es comparar el comportamiento esperado del software con su comportamiento real, analizar las diferencias y reportar los resultados.

Roles que participan

- Probador.
- Analista de prueba.
- Desarrollador del software.

Entradas

- Casos de prueba.
- Datos de prueba.
- Estrategia de prueba.
- Componentes de prueba.
- Configuración del entorno de prueba.

Actividades

- Ejecutar los casos de pruebas.
- Identificar los errores obtenidos.
- Determinar los resultados de las pruebas.
- Registrar los defectos encontrados.

Salidas

- Artefacto Test Script.
- Artefacto Test Log.
- Artefacto Resultado de la prueba.

Representación Gráfica

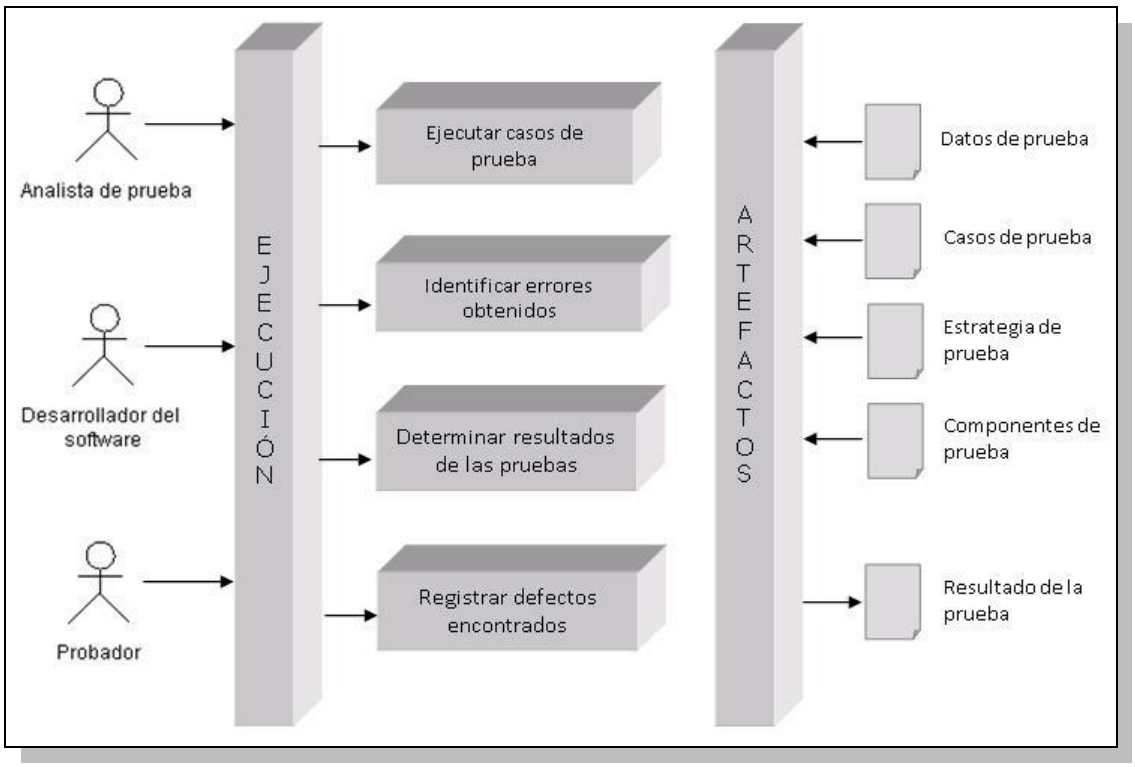


Figura 13: Ejecución de las pruebas

2.5.5. Evaluación de las pruebas

Al obtener los resultados de las pruebas se hace una comparación de estos con los resultados esperados, con el objetivo de verificar si las pruebas fueron factibles o no.

Objetivo

El propósito de este proceso es evaluar los resultados obtenidos y elaborar un documento con los errores detectados, así como recomendaciones y observaciones para futuras pruebas a realizar.

Roles que participan

- Administrador de Pruebas.
- Responsable de calidad interna.

Entradas

- Resultado de la prueba.

Actividades

- Evaluar la cobertura de los casos de pruebas.
- Analizar los defectos.
- Elaborar el informe de evaluación.

Salidas

- Artefacto Sumario de evaluación de las pruebas.

Representación Gráfica

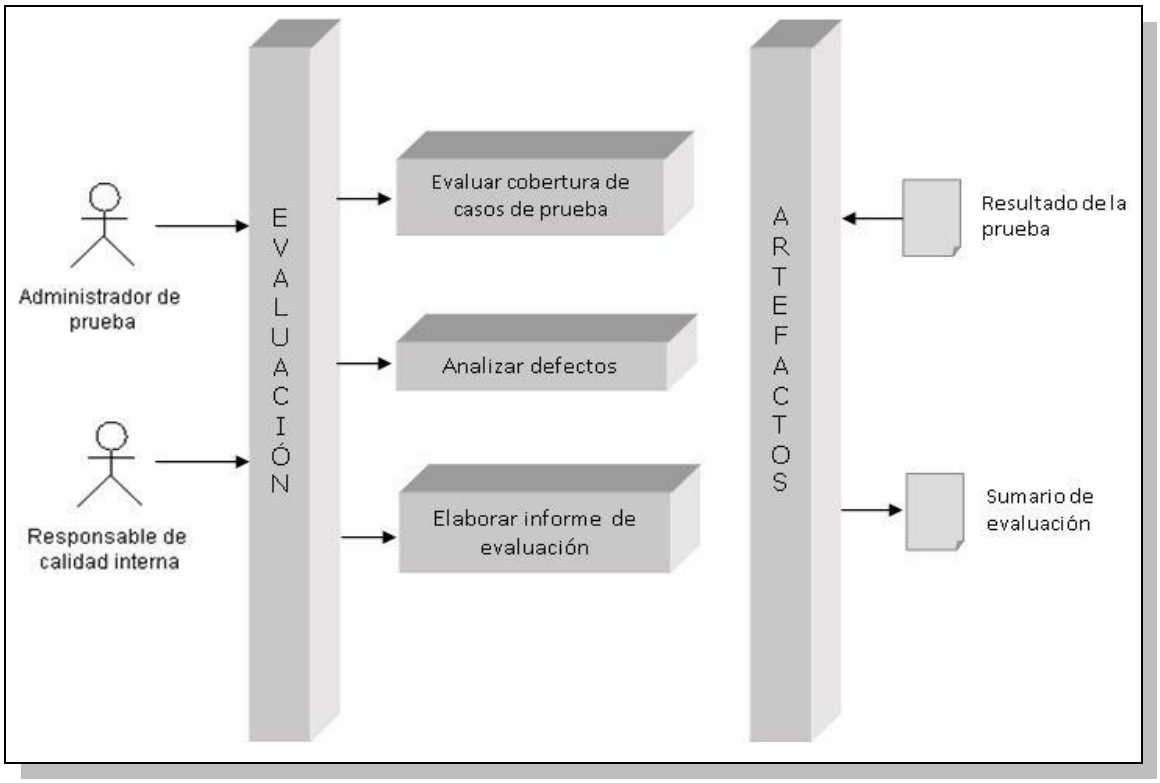


Figura 14: Evaluación de las pruebas

Para el análisis de los defectos encontrados estos deben clasificarse según la prioridad (urgente, alto, normal) y según el grado de severidad (crítico, alto, bajo). (Ver Anexo 6)

Durante todo el ciclo de desarrollo de las pruebas el Administrador de pruebas junto al Responsable de calidad interna deben realizar dos actividades fundamentales: la evaluación y auditoría de la calidad, y la evaluación y mejora de las pruebas.

2.6. Elementos que contribuyen a un mejor proceso de pruebas

Para obtener un buen resultado a partir de cada uno de los procesos de prueba descritos en el epígrafe anterior, es necesario tener en cuenta un conjunto de elementos. Estos juegan un importante papel ya que optimizan el tiempo y el esfuerzo, lo cual nos permite realizar un mayor número de pruebas y de esta forma encontrar la mayor cantidad de errores posibles. Estos son:

- Herramientas
- Métricas

2.6.1. Herramientas

Las pruebas unitarias se pueden efectuar de forma manual, pero cuando se van a aplicar a proyectos de gran magnitud esto se vuelve muy engorroso. En ocasiones, se hace muy difícil ejecutar un gran número de pruebas dado el factor tiempo, que es muy importante para el desarrollo total del proyecto. Por esto se considera que es una buena práctica la automatización de las pruebas para cualquier proyecto de software. En el proyecto Registros y Notarías, se decidió emplear la herramienta NUnit que se ajusta a la plataforma .Net para llevar a cabo dicho proceso.

NUnit es un framework open source de Pruebas de unidad para Microsoft .NET. Tiene la misma funcionalidad que JUnit, y es uno de muchos en la familia xUnit.

NUnit.Forms y NUnit.ASP constituyen una expansión al framework núcleo NUnit. Estos tienen como objetivo ampliar NUnit de manera que este sea capaz de manejar pruebas de elementos de interfaz de usuario tanto en Windows Forms como en ASP.NET.

NUnit posee un conjunto de meta atributos y aserciones que permiten probar los métodos de una clase especificada. Se puede ejecutar desde la consola o a través de una interfaz gráfica y se puede integrar con el Visual Studio en cualquiera de sus versiones. Actualmente soporta los frameworks 1.1/2.0.

La automatización de las pruebas trae numerosas ventajas, pues es una garantía para la calidad del producto a desarrollar. Es por esto que se recomienda llevar a cabo dicho proceso siempre que sea posible. Las pruebas automatizadas se hacen repetibles, optimizan en gran medida el tiempo, lo que es fundamental y muy positivo en el proceso de desarrollo del software; fomentan el cambio, ya que permiten probar cambios en el código y se aseguran de que en ellos no se hayan introducido errores funcionales; simplifican la integración; ya que permiten llegar a la fase de integración con un alto grado de seguridad sobre el código; documentan el código; separan la

interfaz y la implementación; los defectos están acotados y son fáciles de localizar. De cierta forma también le permiten al desarrollador pensar como el consumidor del código y no como el productor, lo que son puntos favorables para el desarrollo del proceso de pruebas.

2.6.2. Métricas de calidad

Una métrica es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado (22).

Se definen entonces como métricas de software a: "La aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo del software y sus productos, para suministrar información relevante a tiempo, así el administrador junto con el empleo de estás técnicas mejorará el proceso y sus productos" (22).

Se considera que las métricas de software nos permiten estimar costo y esfuerzo en realizar un proyecto, evaluar la calidad del software realizado, predecir el tiempo invertido en mantenimiento de un programa o validar las mejores prácticas para el desarrollo del mismo.

Las métricas de calidad definen de una u otra forma la calidad del software; tales como exactitud, estructuración o modularidad, pruebas, mantenimiento, reusabilidad, entre otras. Estas son los puntos críticos en el diseño, codificación, pruebas y mantenimiento. Entre las métricas de calidad se encuentran las siguientes:

Métricas de complejidad

Entre las métricas de complejidad más útiles en la Ingeniería de Software se encuentra la complejidad ciclomática, que proporciona una medición cuantitativa de la complejidad lógica de un programa. Esta métrica permite apreciar la calidad del diseño de un software, así como su grado de comprensibilidad; y además, nos da el número de casos de prueba unitarios básicos para obtener una cobertura del 100%. El valor calculado como complejidad ciclomática, tratado en el capítulo 1, define el número de caminos independientes de un programa y nos da un límite superior para el número de pruebas que se deben realizar asegurando que se ejecuta cada sentencia al menos una vez.

Cobertura de las pruebas

Las métricas de cobertura de las pruebas, indican cómo se van cumpliendo los casos de prueba especificados; en este caso, mientras mayor sea la cobertura, mayor número de casos de prueba

se estarán cumpliendo. De esta manera se lleva un control del cumplimiento de estos para cubrir los requisitos, dando una medida de cómo se está desarrollando el proceso de prueba.

Madurez de las pruebas

Esta métrica es un indicador de cómo se está desarrollando el proceso de pruebas, no solo se preocupa de la completitud de los casos de prueba según los definidos para cumplir los requisitos, sino que también se interesa por cuales han obtenido resultados satisfactorios, para ello es necesario llevar un control de los casos de prueba que arrojaron resultados satisfactorios y el total de los casos de prueba definidos para el cumplimiento de los requisitos.

Porciento del tiempo total dedicado a las pruebas

Esta métrica proporciona una medida del porciento del tiempo dedicado a las pruebas respecto al tiempo total del proyecto. El tiempo de la prueba será mayor mientras más defectos se hayan introducido en el software. Este tiempo dedicado a las pruebas dependerá en gran medida del tamaño y complejidad del software que se esté desarrollando.

Métricas EED

Eficacia de la Eliminación de Defectos

EED = E/ (E+D) donde

E = número de errores encontrados antes de la entrega del software al usuario final.

D = número de errores encontrados después de la entrega al usuario final.

Si el número de defectos presentados después de la entrega al usuario final es 0, entonces el valor de EED es 1.

2.7. Conclusiones

En este capítulo se dio cumplimiento al objetivo general planteado; se elaboró la estrategia la cual se rige por un conjunto de procesos que se definieron. Para el desarrollo de la misma se propusieron niveles de pruebas por los que transitarán los procesos que forman parte de la estrategia, aplicando las técnicas de Caja Blanca y Caja Negra según correspondan.

Esta estrategia se apoya fundamentalmente en la metodología RUP, pues es favorable para proyectos de gran magnitud como lo es el proyecto Registros y Notarías. Se realizó la selección de los roles que se considera deben intervenir en cada uno de los procesos, así como los artefactos

necesarios para que estos se encuentren debidamente documentados, dando paso a cada una de las actividades a desarrollar.

Se procedió a la selección de la herramienta a emplear para la automatización de las pruebas, específicamente para la automatización de las pruebas de Caja Blanca, así como la selección de las métricas que se emplean para evaluar la completitud del proceso.

Capítulo 3: Aplicación de la estrategia

3.1. Introducción

En este capítulo se lleva a cabo la estrategia definida en el módulo Requisiciones del sistema Administración Financiera del proyecto Registros y Notarías. Para esto se realiza la planificación de las pruebas partiendo de las iteraciones definidas para el módulo. En esta actividad se define el esfuerzo a emplear, teniendo en cuenta fundamentalmente los recursos disponibles, tiempo de duración y cantidad de personas involucradas en el proceso, basado en las características y complejidad del producto. Conjuntamente se realiza la configuración del entorno de pruebas, mostrando la distribución de hardware y software necesario para la posterior aplicación de las estas.

Para la realización de las pruebas se analizaron los 5 casos de uso del módulo Requisiciones, en los cuales se identificaron 34 casos de prueba. En este trabajo sólo se expondrán los detalles del caso de uso Registrar Requisiciones en las Unidades Ejecutoras Locales (UEL). En el sumario de evaluación se resumen los resultados obtenidos en el módulo Requisiciones, detallando cuáles son los errores más frecuentes durante la ejecución del sistema. Mediante el empleo de métricas se hace un análisis de la completitud de las pruebas, verificando el porciento de pruebas aplicadas y cuáles de ellas fueron satisfactorias. Para posibilitar el conocimiento de esto por parte de los desarrolladores se hace un breve resumen de los defectos encontrados, su localización, y la prioridad que tienen cada uno de estos.

3.2. Descripción del producto

Actualmente en Venezuela, los Registros Mercantiles y Públicos reciben diariamente muchas peticiones, que sin el control y la tecnología necesaria, el trabajo en estos se hace muy lento y difícil. Para responder a estos problemas se buscó como solución el desarrollo de un sistema informático que permita centralizar y gestionar los procesos registrales, de manera que facilite un mejor servicio y funcionamiento de los Registros y Notarías de la República Bolivariana de Venezuela según lo estipulado en la Ley de Registros Público y del Notariado.

Requisiciones es uno de los módulos del sistema Administración Financiera, perteneciente al proyecto. En este módulo es donde se registran las requisiciones enviadas por las Unidades Ejecutoras Locales (UEL) a sus Unidades Administradoras (UA) que no son más que las

solicitudes de lo bienes y servicios que estas necesitan. A su vez, las Unidades Administradoras Desconcentradas (UAD) gestionan dichas requisiciones dándole solución a cada uno de lo bienes o servicios incluidos en la solicitud, enviándolos a los destinos correspondientes ya sea compras, servicios, almacén, o solicitarlos a la Unidad Administradora Central (UAC) en caso de que no puedan darle solución.

El sistema desarrollado esta compuesto por dos secciones fundamentales:

- Gestión de Requisiciones en la UAD y UAC: Permite anular o aprobar una Requisición cambiando el estatus de esta en cada operación; modificar y visualizar un reporte con los datos de una Requisición; visualizar los datos de la solicitud de la requisición enviada por la UAD o las UEL y gestionar la solución de cada uno de los bienes o servicios solicitados en la misma. También posibilita la realización del desglose de la cantidad que corresponde realizar por la petición de entrega en almacén y la cantidad que se solicita comprar; verificar la disponibilidad de un bien en el almacén, seleccionar el bien o servicio y aplicar estatus.
- Consulta y Registro en la UEL, UAD y UAC: Permite crear, modificar o eliminar una Requisición; autorizar una Requisición si esta se encuentra en edición; visualizar un reporte con los datos de la Requisición o un reporte con el Registro de las Requisiciones existentes. Posibilita la consulta del catálogo de bienes y servicios para agregarle un bien o servicio a una Requisición determinada.

3.3. Plan de pruebas

3.3.1. Propósito

El plan de pruebas del módulo Requisiciones del proyecto Registros y Notarías consiste en seleccionar y coordinar todas las actividades para garantizar la calidad del producto durante el ciclo de vida del proyecto. En el mismo se explica alcance, requerimientos a probar, estrategia a probar, recursos requeridos, calendario, herramientas así como los responsables involucrados en el proceso de pruebas y se mencionan los entregables que son salidas generadas en cada una de las actividades.

3.3.2. Alcance

En el módulo Requisiciones del proyecto Registros y Notarías durante el proceso de pruebas se llevarán a cabo las pruebas de Unidad, Integración y Validación, aplicando las técnicas de prueba de Caja Blanca y Caja Negra. Las mismas comprobarán el correcto funcionamiento del software tanto internamente como la parte externa del producto. Estas pruebas deben realizarse en el orden en que aparecen, luego de que se hayan corregido los errores encontrados en la etapa anterior.

1- Pruebas de Unidad: se realiza a través de la técnica de prueba de Caja Blanca, con el objetivo de comprobar el funcionamiento interno de cada unidad, siendo esta la escala más pequeña que se puede probar.

2- Pruebas de Integración: se realiza mediante las técnicas de prueba de Caja Blanca y Caja Negra. Una vez realizada la prueba de Unidad pasan a integrarse todas estas unidades probadas anteriormente para comprobar su funcionamiento como un sistema ya conformado.

3- Pruebas de Validación: se realiza a través de la técnica de prueba de Caja Negra, para verificar que las funcionalidades probadas responden a los requerimientos establecidos.

4- Pruebas de Sistema: Es recomendable la realización de las pruebas de Caja Negra además de las otras pruebas que se llevan a cabo en esta etapa.

3.3.3. Referencias

Documento de Requerimientos de Requisiciones.
Modelo de Casos de Uso del Sistema Requisiciones.
Prototipo de la Interfaz de Usuario Requisiciones.
Pautas de Requisiciones.
Glosario de términos.
Documento Visión.

3.3.4. Requerimientos a probar

Gestionar Registro de Requisiciones.

- Realizar una búsqueda de las requisiciones por código de la UEL o denominación de la UEL.

- Realizar una búsqueda de las requisiciones por el tipo de requisición, un intervalo de fecha y un estatus.
- Crear una requisición.
    - Mostrar los siguientes datos para que los mismos sean editados o seleccionados:
        - ✓ Tipo de requisición.
        - ✓ Fecha.
        - ✓ Observación.
        - ✓ Código del Bien o Servicio.
        - ✓ Unidad de medida.
        - ✓ Precio.
        - ✓ Cantidad.
    - Mostrar los proyectos o acciones centralizadas para seleccionarlas por cada Bien o Servicio.
    - Realizar una búsqueda del Bien o Servicio que se desee.
    - Visualizar un reporte donde se muestre por cada Bien o Servicio la asignación presupuestaria por Proyecto o Acciones Centralizadas y Acciones Específicas.
- Eliminar una requisición.
- Modificar una requisición.
- Autorizar una requisición si la misma se encuentra en estado de edición.
- Visualizar un reporte de las requisiciones que han sido creadas donde se muestre por estatus de la requisición, el Nro., tipo de requisición, observación, código de la UEL, fecha y monto.

Gestionar Requisiciones.

- Realizar una búsqueda de los tipos de requisiciones por el tipo de requisición, un intervalo de fecha, el código de la UEL o la denominación de la UEL y un estatus.
- Visualizar los datos de la requisición.
    - Mostrar todos los Bienes/Servicios que fueron solicitados por la requisición.

- Visualizar un reporte que muestre por cada requisición los Bienes o Servicios que fueron solicitados con la asignación presupuestaria por Proyecto o Acción Centralizada y Acciones Específicas y el estatus de la misma.

- Aprobar una requisición si la misma ya fue solicitada.
- Anular una requisición si la misma ya fue solicitada.
- Visualizar un reporte donde se muestre en la UAD las requisiciones por cada una de las UEL.

Gestionar Reportes.

- Visualizar reporte.
- Imprimir el reporte.
- Exportar el reporte a formato PDF, XLS y DOC.

3.3.5. Estrategia de pruebas

En la estrategia de pruebas se definen los tipos de pruebas a realizar, las técnicas a emplear, así como el criterio de evaluación de las mismas.

Para el módulo Requisiciones se planificaron los siguientes tipos de prueba:

3.3.5.1. Tipos de prueba

Nivel de unidad

Objetivos	Someter a pruebas a la menor unidad del programa para determinar si está correctamente implementada, de acuerdo con la especificación del sistema.
Técnicas	Emplear la técnica de Caja Blanca mediante el método del camino básico para la obtención de los casos de prueba y la herramienta NUnit para su ejecución.
Criterio de completitud	Las pruebas planeadas han sido ejecutadas. Todos los errores identificados han sido corregidos.

Nivel de integración

Objetivos	Verificar que los módulos integrados funcionan correctamente
Técnicas	Emplear la técnica de Caja Negra a través de los métodos de partición equivalente y análisis de valores límites y la técnica de Caja Blanca mediante el método del camino básico
Criterio de completitud	Las pruebas planeadas han sido ejecutadas. Todos los defectos identificados han sido corregidos.

Nivel de validación

Pruebas funcionales

Objetivos	Verificar que el sistema responde a los requisitos funcionales especificados.
Técnicas	Emplear la técnica de Caja Negra a través de los métodos de partición equivalente y análisis de valores límites.
Criterio de completitud	Las pruebas planeadas han sido ejecutadas. Todos los defectos identificados han sido corregidos.

Nivel de sistema

Objetivos	Verificar que las funcionalidades del sistema obtenido respondan correctamente a los requisitos funcionales especificados. Verificar la entrada y salida de los datos.
Técnicas	Emplear la técnica de Caja Negra a través de los métodos de partición equivalente y análisis de valores límites.
Criterio de completitud	Las pruebas planeadas han sido ejecutadas. Todos los defectos identificados han sido corregidos.

3.3.5.2. Criterio de evaluación

Con el objetivo de evaluar el comportamiento de las pruebas, se empleará la métrica de cobertura de pruebas basada en los requerimientos del software. Esta indica cómo se van cumpliendo los casos de prueba especificados, por lo tanto mientras mayor sea la cobertura, mayor número de casos de prueba se estarán cumpliendo.

De esta forma se analizan los resultados de las pruebas ejecutadas, verificando cuales fueron satisfactorias con respecto a los requerimientos definidos en el plan.

Además se empleará la métrica de la cobertura de la prueba basada en el código, para obtener una medida de cuántas líneas han sido cubiertas por las pruebas, haciendo una comparación entre el código ejecutado por la prueba y el código que no ha sido ejecutado.

3.3.6. Herramientas

Para la realización de las pruebas es necesario el empleo de un conjunto de herramientas que posibilitarán que las mismas se apliquen con mayor eficiencia y rapidez.

	Herramientas	***Versión***
Administración del proyecto	Microsoft Project Microsoft Word	2003 2007
Configuración del entorno de prueba	Microsoft Visio	2007
Pruebas de unidad	NUnit	Framework 2.0

3.3.7. Recursos

Recursos Humanos

Rol	***Recursos mínimos***	***Responsabilidades***

Administrador de pruebas	1	• Define el plan de pruebas. • Realiza la evaluación final de las pruebas.
Analista de pruebas	2	• Definir los casos de prueba y datos de prueba. • Define los procedimientos de prueba.
Diseñador de pruebas	1	• Define la estrategia de prueba. • Realiza la configuración del entorno de pruebas.
Probador	2	• Ejecuta las pruebas. • Elabora el documento con los errores encontrados.
Responsable de calidad interna	1	• Revisa los artefactos a entregar. • Planifica cursos de capacitación al personal. • Realiza el informe único de No Conformidades.
Desarrollador	2	• Apoyar el diseño de los casos de prueba de Caja Blanca. • Apoyar la implementación y ejecución de las pruebas de unidad.

Recursos del sistema

Recursos	***Comentario***
Repositorio para las pruebas	Servidor para la seguridad y seguimiento de las pruebas.

6 UPS	Una UPS (sistema de energía ininterrumpida) para cada computadora.
6 PC Clientes	Una computadora personal para cada integrante del equipo.
1 Impresora	Importante para la impresión de los trámites gestionados por el sistema.
1 Escáner	Importante para el escaneo de los documentos necesarios para el funcionamiento adecuado del sistema.

3.3.8. Hitos del Proyecto

A continuación se muestra el cronograma de realización de cada una de las actividades pertenecientes a los procesos de Planificación, Diseño, Implementación, Ejecución y Evaluación de las pruebas en su primera iteración para el módulo Requisiciones.

Procesos	***Actividades***	***Inicio***	***Fin***	***Duración***
Planificación	Identificar objetivos de las pruebas.	17/04/08	18/04/08	2
	Evaluar recursos necesarios.	18/04/08	19/04/08	2
	Definir los tipos de prueba a realizar.	19/0408	19/04/08	1
	Definir las pautas de las pruebas.	19/04/08	21/04/08	2
	Definir la configuración del entorno de prueba.	21/04/08	21/04/08	1
Diseño	Identificar los datos de prueba.	22/04/08	24/04/08	3
	Identificar y describir los casos de prueba.	24/04/08	30/04/08	7
	Identificar y estructurar los procedimientos de prueba.	30/04/08	2/05/08	3
Implementación	Identificar mecanismos de prueba.	22/04/08	23/04/08	2
	Definir los elementos de prueba a automatizar.	23/04/08	24/04/08	2
	Implementación de las pruebas.	2/05/08	5/05/08	4

Ejecución	Ejecutar los casos de pruebas.	5/05/08	10/05/08	6
	Identificar los errores obtenidos.	10/05/08	10/05/08	1
	Registrar los defectos encontrados.	10/05/08	11/05/08	2
	Determinar los resultados de las pruebas.	11/05/08	11/05/08	1
Evaluación	Evaluar la cobertura de los casos de pruebas.	12/05/08	12/05/08	1
	Analizar los defectos.	12/05/08	12/05/08	1
	Elaborar el informe final de evaluación de las pruebas.	12/05/08	13/05/08	2

3.3.9. Entregables

A continuación se relacionan los entregables más importantes durante todo el proceso de pruebas. Cada uno de ellos constituye el resultado de una actividad.

- Plan de pruebas
- Entorno de Configuración de pruebas
- Datos de prueba
- Casos de prueba
- Documento de No Conformidades
- Sumario de evaluación de las pruebas

3.4. Configuración del entorno de pruebas

Un entorno de prueba proporciona una respuesta adecuada y controlada para llevar a cabo las actividades necesarias de evaluación. Este constituye un aspecto importante para lograr eficacia en el análisis de fallos y en la reparación de errores.

Para la ejecución satisfactoria de las pruebas es necesario realizar adecuadamente la selección del software y hardware requerido. Para ello es importante saber que el entorno de pruebas debe estar separado físicamente del entorno de producción.

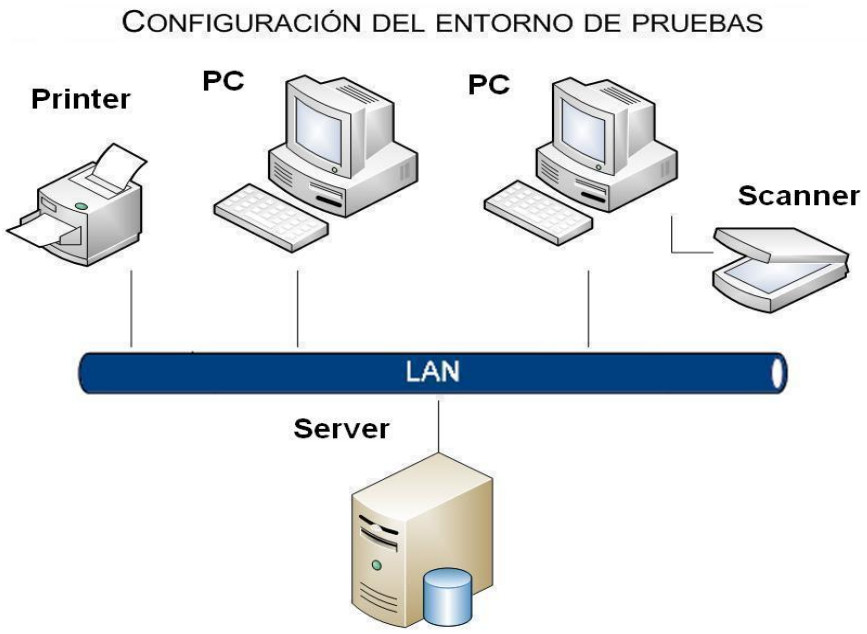


Figura 15: Configuración del entorno de prueba

La siguiente tabla recoge los datos del entorno de pruebas obtenido para el sistema Requisiciones.

Nombre	***Entorno de prueba para el sistema Requisiciones***
Propósito	Con la configuración del entorno de pruebas se crea un ambiente próximo al real, permitiendo que el software se pruebe en un ambiente similar al del cliente. Esto permite también desarrollar las pruebas con mayor facilidad, detectando un mayor número de defectos.
Requerimientos de Software	• Sistema Operativo en las PC: Windows XP SP 2 Español • Sistema Operativo en los Servidores: Red Hat AS 4.0 Enterprise Edition y Windows 2003 SBS Español • Scanner: LeadTools Raster Image Pro for .NET version 14.5 • Office 2003 Español • Framework de .NET versión 1.1
Requerimientos de Hardware	• Procesador: Intel® Pentium® 4 con tecnología Hyper Threading (de 3,0 GHz, 1 o 2 MB de caché de nivel 2) • Tipo de memoria: 512 MB DDR2-Synch DRAM PC2-3200 y PC2-

	4200 • Unidad de disco duro: 80 GB SATA • Tarjeta de red: Intel Pro 1000 MT Gigabit NIC.
Procedimiento para la recuperación y restauración del entorno de pruebas.	Crear un backup de la Base de Datos diariamente y realizar copias de seguridad en cada equipo. Esto posibilitará que dada cualquier dificultad, la configuración del entorno de pruebas pueda ser recuperada y restaurada, minimizando el tiempo y esfuerzo, y facilitando el trabajo.

De acuerdo a las etapas definidas en el alcance del plan, las pruebas a realizar en el módulo Requisiciones estarán centradas en las pruebas de integración mediante las técnicas de Caja Blanca y Caja Negra, debido a que el módulo se encuentra actualmente en la integración de sus casos de uso. Para las pruebas de Caja Negra se diseñaron datos de prueba que permitan comprobar cada una de las respuestas que pueda dar el sistema. Estos se detallan a continuación.

3.5. Datos de prueba

Para efectuar las pruebas de Caja Negra, es necesario confeccionar una colección de datos que sirvan de entrada para obtener todas las posibles respuestas del sistema.

Para ello es necesario usar datos que seguramente serán empleados por los usuarios, e identificar qué rangos de datos pueden alterar el comportamiento del programa, tratando de esforzar este al máximo. Según (23) la experiencia indica, además, que suelen producirse fallos en los bordes de las zonas, por lo que se recomienda probar siempre con datos extremos.

Para la confección de datos de prueba que permitan que la funcionalidad del caso de prueba sea la correcta y se corresponda con los requisitos del cliente para el módulo, se deben tener en cuenta algunas especificaciones como:

- Las Fechas deben mostrarse en el formato: día/mes/año.

  Ejemplo: (dd/mm/aaaa)

- Los componentes relacionados con Nombres, Descripciones, Denominaciones, Observaciones, que son de caracteres deben permitir la entrada de números y los símbolos: # - , /: ( ).
- La Cantidad es un componente numérico con dos decimales.

  Ejemplo: 1.000.000.000,00

Teniendo en cuenta la técnica de prueba escogida, se diseñaron un conjunto de datos necesarios para efectuar las pruebas. En este caso, se escogieron datos para probar que la funcionalidad es correcta y datos que no son admisibles para lograr detectar el mayor número de errores posibles en cada uno de los campos.

Datos de prueba para el caso de uso: <Registrar Requisiciones en la UEL>

Fecha desde	Fecha hasta	Cantidad (Bien o Servicio)	Cantidad (Proy/AC)	Observaciones
03/05/2008	03/06/2008	1.300.400.00	1.500.600.00	Prueba
40/32/3000	03/06/2008	1.300.400.00	1.500.600.00	Prueba
03/05/2008	32/40/2050	1.300.400.00	1.500.600.00	Prueba
03/05/2008	03/06/2008	1a\;000.000	1.500.600.00	Prueba
03/05/2008	03/06/2008	1.300.400.00	200.*a$.000	Prueba
03/05/2008	03/06/2008	1.300.400.00	1.500.600.00	P1;$@ 123 va

3.6. Procedimiento de prueba

3.6.1. Procedimiento para las pruebas de Caja Negra

Para el caso de uso Registrar Requisiciones en la UEL se han defino una serie de procedimientos que son la base para la realización del diseño de los casos de prueba de Caja Negra. Estos guiarán el proceso puesto que son los que describen de forma general como se debe ejecutar un caso de prueba. La siguiente tabla muestra el procedimiento de prueba para el caso de prueba particular Registrar Requisiciones. Los demás procedimientos del caso de uso pueden se pueden ver en el Anexo #2

Caso de uso: <Registrar Requisiciones en la UEL>

Procedimiento: “Registrar Requisición”

No	Prueba	Acciones a realizar	Resultado esperado
1	Registrar Requisición.	Seleccione en el menú la opción Consulta y Registro, luego la opción Requisiciones en la UEL. Seleccione la opción Crear y seleccione el Tipo de Requisición, adicione el Bien o Servicio modificando la Cantidad. Luego debe adicionar el Proy/AC y debe introducir las observaciones necesarias.	Se verifica que los datos seleccionados e introducidos son correctos y se crea una nueva Requisición, mostrándose esta en la interfaz Consulta y Registro de Requisiciones.
1.1	Registrar Requisición.	Realice una selección no válida de datos o introduzca datos erróneos.	El sistema debe mostrar un mensaje de error.

3.6.2. Procedimiento para las pruebas de Caja Blanca

Para la ejecución de las pruebas de Caja Blanca se utilizará la herramienta NUnit. A continuación se hará una descripción de cómo emplear esta herramienta.

Primeramente se crea una nueva capa asociada a la solución para implementar los casos de prueba de Caja Blanca.

Se debe agregar la librería NUnit.Framework.

Antes del nombre de la clase se debe poner [TestFixture] que es el atributo que utiliza NUnit para indicar que la clase contiene código de prueba. Esta clase es pública y cuenta con un constructor que provee el framework.

Todos los métodos de esta clase deben poseer el atributo [Test], que se utiliza para marcar un método como método de prueba. Estos siempre deben retornar *void* y no reciben parámetros.

Además de basar su funcionamiento en atributos personalizados, NUnit utiliza aserciones, que no son más que métodos del framework de NUnit, utilizados para comprobar y comparar valores.

3.7. Casos de prueba

Los casos de pruebas deben comprobar que el producto que se encuentra en desarrollo, satisface realmente las peticiones del usuario final, tal y como se describe en la especificación de los requerimientos y los casos de uso.

Es importante destacar la necesidad de confeccionar un buen caso de prueba, pues de esta forma se detectarán la mayor cantidad de errores posibles, permitiendo con esto que el software llegue libre de defectos a las manos del usuario final.

3.7.1. Casos de prueba de Caja Negra para el módulo Requisiciones.

Para el caso de uso Registrar Requisiciones en la UEL del módulo Requisiciones se diseñaron quince casos de prueba. En la siguiente tabla se muestra como quedó diseñada una de las secciones que cuenta con un caso de prueba. Los demás casos de prueba se pueden ver en el Anexo #3.

Caso de prueba Registrar Requisiciones en la UEL

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC1: Registrar Requisiciones	EC 1.1: Flujo Básico de Registrar Requisiciones	El sistema permite crear una Requisición	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados 6. El usuario selecciona la opción Crear Requisición.

			7. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 8. El usuario selecciona el tipo de requisición y edita la observación de la misma. 9. El usuario selecciona el Bien o Servicio. 10. El usuario edita el dato cantidad. 11. El usuario selecciona Proyecto/Acción Centralizada y la Acción Específica del Bien o Servicio antes seleccionado. 12. El usuario selecciona la opción Aceptar. 13. El sistema Muestra interfaz Registro de Requisiciones con lista actualizada.
	EC 1.2: Flujo Alterno Cerrar	El sistema permite cerrar la interfaz sin realizar cambios.	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 6. El usuario selecciona la opción Cerrar. 7. El sistema no realiza ningún cambio. Cierra la interfaz

			correspondiente. Terminando así el caso de uso.
	EC 1.3: Flujo Alterno Cancelar	El sistema permite cancelar la interfaz sin realizar cambios.	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados 6. El usuario selecciona la opción Crear Requisición. 7. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 8. El usuario selecciona el tipo de requisición y edita la observación de la misma. 9. El usuario selecciona el Bien o Servicio. 10. El usuario edita el dato cantidad. 11. El usuario selecciona Proyecto/Acción Centralizada y la Acción Específica del Bien o Servicio antes seleccionado. 12. El usuario selecciona la opción Cancelar. 13. El sistema no realiza ningún cambio. Cierra la interfaz correspondiente.

SC2: Eliminar Requisición	EC 2.1: Flujo Básico Eliminar Requisición	Permite eliminar una Requisición que esté en estado de edición.	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 6. El usuario selecciona una requisición y ordena Eliminar Requisición. 7. El sistema comprueba que la Requisición se encuentre en estado de edición, si es así se elimina.
	EC 2.2: Flujo Alterno Eliminar Requisición	El sistema no permite eliminar una Requisición que no esté en estado de edición.	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 6. El usuario selecciona una requisición y ordena Eliminar Requisición. 7. El sistema comprueba que la

			Requisición se encuentre en estado de edición, si es así no la elimina.
SC3: Modificar Requisición	EC 3.1: Flujo Básico Modificar Requisición	El sistema permite modificar una Requisición.	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 6. El usuario selecciona la opción Modificar Requisición 7. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 8. El usuario selecciona el tipo de requisición y edita la observación de la misma. 9. El usuario selecciona el Bien o Servicio. 10. El usuario edita el dato cantidad. 11. El usuario selecciona Proyecto/Acción Centralizada y la Acción Específica del Bien o Servicio antes seleccionado. 12. El usuario selecciona la opción Aceptar.

			13. El sistema Muestra interfaz Registro de Requisiciones con lista actualizada.
	EC 3.2: Flujo Alterno Modificar Requisición	El sistema permite cancelar la modificación realizada	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 6. El usuario selecciona la opción Modificar Requisición 7. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 8. El usuario selecciona el tipo de requisición y edita la observación de la misma. 9. El usuario selecciona el Bien o Servicio. 10. El usuario edita el dato cantidad. 11. El usuario selecciona Proyecto/Acción Centralizada y la Acción Específica del Bien o Servicio antes seleccionado. 12. El usuario selecciona la opción Cancelar.

			13. El sistema no realiza ningún cambio. Cierra la interfaz correspondiente.
SC4: Autorizar Requisición	EC 4.1: Flujo Básico Autorizar Requisición	Permite autorizar la requisición si está en estado de edición	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 6. El usuario selecciona la opción Autorizar. 7. El sistema verifica que la requisición esté en estado de edición. 8. El sistema muestra interfaz Autorización de la Requisición donde se confirma la autorización de la requisición. 9. El usuario selecciona la opción Aceptar. 10. El sistema muestra interfaz Registro de Requisiciones con cambios en el estado de las requisiciones.

	EC 4.2: Flujo alterno Cancelar	Permite cancelar la operación	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 6. El usuario selecciona la opción Autorizar. 7. El sistema verifica que la requisición esté en estado de edición. 8. El sistema muestra interfaz Autorización de la Requisición donde se confirma la autorización de la requisición. 9. El usuario selecciona la opción Cancelar. 10. El sistema no realiza ningún cambio. Cierra la interfaz correspondiente.

3.7.2. Casos de prueba de Caja Blanca para el módulo Requisiciones.

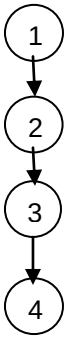
Para el caso de uso Registrar Requisiciones en la UEL del módulo Requisiciones se diseñaron un conjunto de casos de prueba de Caja Blanca mediante la técnica del camino básico. A continuación se muestran algunos de estos casos de prueba.

Caso de uso Registrar Requisiciones en la UEL

Método

public ArrayList BuscarRequisicion (int idTipoRequisicion, DateTime fechaDesde, DateTime fechaHasta, int idEstado)

Grafo de flujo



Complejidad ciclomática

V (G) = 3 - 4 + 2 = 1

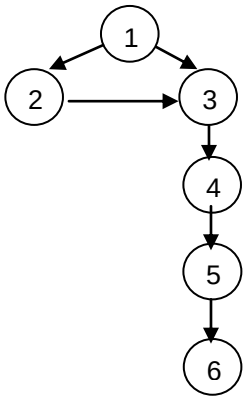
Caminos Independientes

Camino 1: 1 – 2 – 3 - 4

Método

public ArrayList ObtenerCatalogoBienesServicios (string descripcion)

Grafo de flujo



Complejidad ciclomática

V (G) = 6 – 6 + 2 = 2

Caminos Independientes

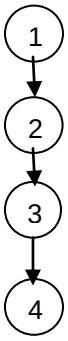
Camino 1: 1 – 2 – 3 – 4 – 5 - 6

Camino 2: 1 – 3 – 4 – 5 - 6

Método

public ArrayList ObtenerPeticionesRequicision (decimal idRequicision)

Grafo de flujo



Complejidad ciclomática

V (G) = 3 - 4 + 2 = 1

Caminos Independientes

Camino 1: 1 – 2 – 3 - 4

Tras haber identificado los casos de prueba de Caja Blanca para el caso de uso Registrar Requisiciones en la UEL, se procedió a su ejecución obteniendo los resultados que se muestran en el siguiente epígrafe.

3.8. Sumario de evaluación de las pruebas

3.8.1. Resumen de los resultados obtenidos

Para la ejecución de las pruebas de Caja Negra se contó con un total de 19 requisitos funcionales.

Según los requerimientos definidos en el plan de pruebas, la cobertura de pruebas planificadas fue completada al 100 %.

Con las pruebas que fueron ejecutadas, según lo planificado, se cubrió el 94 % de los requisitos funcionales.

De las pruebas planificadas y ejecutadas solo un 47 % fueron satisfactorias.

En la ejecución de las pruebas de Caja Blanca se probaron las principales funcionalidades del caso de uso Registrar Requisiciones en la UEL correspondiente al módulo Requisiciones. De 4 casos de prueba se obtuvieron 3 satisfactorios y 1 caso de prueba fallido (Ver Anexo 5).

3.8.2. Cobertura de las pruebas

Para obtener la cobertura de las pruebas de Caja Negra que fueron planificadas y ejecutadas en el módulo Requisiciones se recogieron los siguientes resultados:

- Casos de prueba diseñados: 53
- Casos de prueba aplicados: 51
- Casos de prueba satisfactorios: 25
- Cobertura de pruebas (ejecutadas) =51/53 = 96%
- Cobertura de pruebas (exitosas)= 25/51= 49%

En la ejecución de las pruebas de Caja Blanca se obtuvo:

- Cobertura de pruebas (ejecutadas) = 100%

- Cobertura de pruebas (exitosas)= 3/4= 75%

3.8.3. Análisis de los defectos encontrados

A continuación se muestran los principales defectos encontrados en el caso de uso Registrar Requisiciones en la UEL durante la ejecución de los casos de prueba de Caja Negra.

Nro.	No Conformidad	Aspecto correspondiente	Prioridad	Severidad	Estado
1	La opción Tipo de Requisición debe tener los elementos: Todos, Compras y Servicios; en lugar de eso tiene Todos, Tipo 1 y Tipo 2.	Interfaz Consulta y Registro de Requisiciones > Tipo de Requisición	Alta	Alta	Pendiente
2	El sistema permite entra una Fecha Desde mayor que la Fecha Hasta.	Interfaz Consulta y Registro de Requisiciones > Fecha Desde y Fecha Hasta	Alta	Alta	Pendiente
3	El sistema permite escribir en el año de la Fecha Desde y Fecha Hasta cualquier número y no muestra un mensaje de error.	Interfaz Consulta y Registro de Requisiciones > Fecha Desde y Fecha Hasta	Alta	Alta	Pendiente
4	En los campos Tipo de Requisición y Estatus el sistema permite borrar con la tecla Delete.	Interfaz Consulta y Registro de Requisiciones > Tipo de Requisición y Estatus	Normal	Baja	Pendiente

5	En el campo Observación el sistema permite la entrada de un número infinito de caracteres mostrando una excepción.	Interfaz Requisiciones > Observación	Urgente	Crítica	Pendiente
6	En el campo Observación el sistema no permite la entrada de caracteres especiales ni de espacios entre palabras.	Interfaz Requisiciones > Observación	Normal	Baja	Pendiente
7	El sistema muestra una excepción al oprimir el botón Aceptar sin haber entrado ninguna Observación.	Interfaz Requisiciones > Aceptar	Urgente	Crítica	Pendiente
8	El sistema muestra una excepción al tratar de adicionar varios elementos en las regiones Bien o Servicio y Proy/AC.	Interfaz Requisiciones > Bien o Servicio Interfaz Requisiciones > Proy/AC	Urgente	Crítica	Pendiente
9	Cuando se intenta eliminar un Proy/AC el sistema muestra una excepción.	Interfaz Requisiciones > Proy/AC	Urgente	Crítica	Pendiente
10	Al autorizar una requisición el sistema lo actualiza en la lista con un estatus incorrecto.	Interfaz Requisiciones	Alta	Alta	Pendiente

11	El sistema permite autorizar una requisición varias veces (no importa el estado en que se encuentre), en lugar de mostrar un mensaje de error.	Interfaz Consulta y Registro de Requisiciones > Autorizar	Alta	Alta	Pendiente
12	El formato del mensaje de autorización no se corresponde con los demás mensajes.	Interfaz Consulta y Registro de Requisiciones > Autorizar	Normal	Baja	Pendiente
13	Al crear una requisición el sistema actualiza la lista pero todas se agregan con Nro. 0	Interfaz Consulta y Registro de Requisiciones	Normal	Baja	Pendiente
14	El sistema Elimina la Requisición sin mostrar un mensaje de verificación.	Interfaz Consulta y Registro de Requisiciones > Eliminar	Alta	Alta	Pendiente
15	La interfaz Catálogo de Bienes y Servicios no tiene el nombre puesto.	Interfaz Catálogo de Bienes y Servicios	Normal	Baja	Pendiente
16	El campo editable de la interfaz Catálogo no tiene nombre y permite la entrada de un número infinito de caracteres.	Interfaz Catálogo de Bienes y Servicios	Normal	Baja	Pendiente
17	Al oprimir el botón Seleccionar de la interfaz Catálogo el sistema cierra la	Interfaz Catálogo de Bienes y Servicios	Normal	Baja	Pendiente

	interfaz, aunque el sistema no haya encontrado nada en la búsqueda o no haya ningún elemento seleccionado. El sistema debe mostrar un mensaje de error.				
18	El sistema elimina cualquier requisición, no importa en el estado en que se encuentre.	Interfaz Consulta y Registro de Requisiciones > Eliminar	Urgente	Crítica	Pendiente

De los defectos encontrados en el caso de uso Registrar Requisiciones en la UEL tras haber realizado las pruebas de Caja Negra, se identificaron 5 defectos críticos. De acuerdo a la prioridad se identificaron 6 con prioridad alta, 5 con prioridad urgente y 7 con prioridad normal, como se muestra en la siguiente figura:

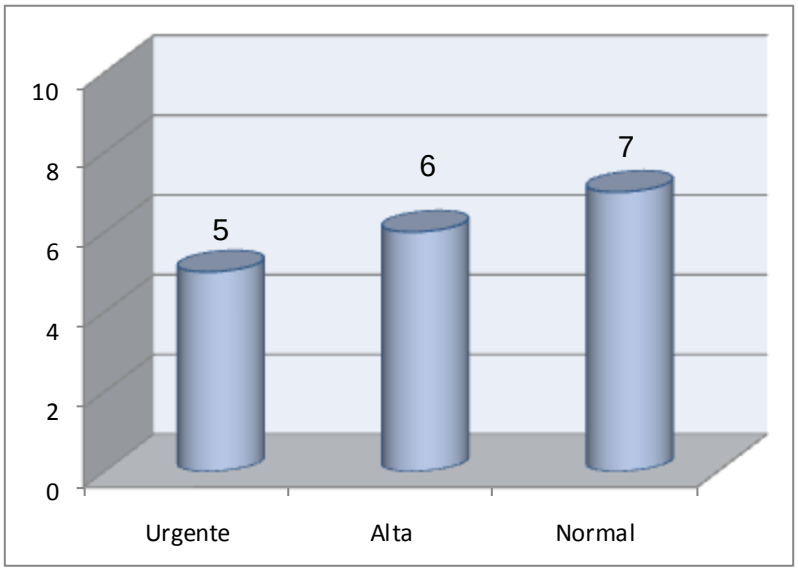


Figura 16: Clasificación de los defectos

3.9. Análisis Comparativo

A continuación se muestra una comparación de los resultados obtenidos luego de haber ejecutado las pruebas siguiendo procedimientos diferentes en los módulos Requisiciones y Contabilidad. Al módulo Contabilidad se le aplicaron las pruebas empleando el procedimiento habitual del equipo de calidad interna del proyecto Registros y Notarías y al módulo Requisiciones se le aplicaron las pruebas con la estrategia propuesta en este trabajo.

Módulo	***Defectos***	***Tiempo (días)***	***Tamaño (CU)***	***Pruebas realizadas***
Requisiciones	18	7	5	Caja Blanca y Caja Negra
Contabilidad	11	16	8	Caja Negra

3.10. Conclusiones

En este capítulo se llevó a cabo la estrategia definida en el módulo Requisiciones del sistema Administración Financiera del proyecto Registros y Notarías. Para esto se siguieron todos los procesos definidos anteriormente en el Capítulo 2. Se realizaron una serie de actividades las cuales tuvieron en cuenta el esfuerzo a emplear, los recursos disponibles, tiempo de duración y cantidad de personas involucradas en el proceso, partiendo de las características y complejidad del producto. Se realizó la configuración del entorno de pruebas donde se muestra la distribución de hardware y software que se emplearon para la realización de las pruebas.

Para la ejecución de las pruebas se diseñaron los datos de prueba necesarios así como los casos de prueba para especificar cada de una de las entradas y condiciones.

Luego de haber aplicado las pruebas se hizo un análisis de los defectos encontrados, clasificándolos según su prioridad para facilitarle el trabajo a los desarrolladores a la hora de darles solución.

Conclusiones Generales

En el presente trabajo se definió una estrategia para el proceso de pruebas en el proyecto Registros y Notarías, la cual se considera mejora dicho proceso. En esta se propuso la aplicación de las técnicas de Caja Blanca y Caja Negra en diferentes niveles de prueba, y se presentaron los roles y artefactos que se consideran fundamentales para que la estrategia tenga el éxito deseado.

La estrategia fue aplicada al módulo Requisiciones del sistema Administración Financiera.

Con la realización de este trabajo se pudo llegar a las siguientes conclusiones:

- El aseguramiento de la calidad del software permite reducir notablemente los costos de producción e implantación del producto, siendo el proceso de pruebas un eslabón fundamental para encontrar la mayor cantidad de defectos posibles y que el producto final llegue a las manos del cliente con la calidad requerida.
- En la estrategia definida se identificaron los procesos que deben guiar las pruebas, así como los roles, artefactos y actividades necesarios para tener un desarrollo satisfactorio de cada una de las fases.
- Los casos de prueba diseñados fueron aplicados mediante las técnicas de prueba de Caja Blanca y Caja Negra, cubriendo la mayor cantidad de combinaciones posibles para la detección de los defectos.
- Al obtener los resultados tras haber ejecutado las pruebas se concluyó que en este primer ciclo el 49 % de las pruebas funcionales realizadas fueron satisfactorias y que se logró probar mediante la herramienta NUnit todo el código del módulo propuesto.
- Se aplicó correctamente el proceso de pruebas para el módulo Requisiciones garantizando, de esta forma, la obtención de un producto aceptable con un número reducido de defectos.

Recomendaciones

Establecer la estrategia propuesta en el proyecto Registros y Notarías y utilizarla como referencia en otros proyectos de desarrollo de software en nuestra universidad.

Previo a la aplicación de la estrategia:

- Lograr que el equipo de desarrollo se involucre y participe en actividades referentes a la calidad, principalmente al proceso de pruebas.
- Realizar una capacitación al personal involucrado en el desarrollo de las pruebas.

Durante la ejecución:

- Realizar encuentros periódicamente con el equipo de pruebas para verificar el cumplimiento de las actividades planificadas.
- Hacer extensivo el uso de la herramienta NUnit en los demás módulos del proyecto para la detección de los errores en el código.
- Llevar a cabo un proceso adecuado para la corrección de los errores por parte de los desarrolladores.

Para ampliar la investigación:

- Realizar un estudio acerca de la automatización de pruebas funcionales y el posible empleo de estas herramientas en el proyecto.
- Continuar la investigación para enriquecer la estrategia, con el objetivo de aportar soluciones novedosas que contribuyan a la calidad del software.

Bibliografía

1. **Carrasco, Oscar M. Fernández.** Un enfoque actual sobre la calidad del software. [Online] diciembre 1995. http://bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm

2. Software Engineering Institute. [Online] [Cited: mayo 5, 2008.] http://www.sei.cmu.edu/cmmi/.

3. International Organization for Standardization. [Online] 2008. [Cited: mayo 5, 2008.] http://www.iso.org/iso/iso_catalogue/management_standards/understand_the_basics.htm.

4. Normas9000. [Online] 2007. [Cited: mayo 6, 2008.] http://normas9000.com/que-es-iso-9000.html.

5. Objetoide. ***Conociendo las pruebas de Software***. [Online] febrero 13, 2008. [Cited: abril 16, 2008.] http://objetoide.blogspot.com/2008/02/conociendo-las-pruebas-de-software.html.

6. **Pressman, R.** *Ingeniería del Software: Un enfoque Práctico.* 2005.

7. IEEE. ***Metrics***. [Online] 1991.

8. **Myers, G.** *The art of software testing.* 1979.

9. **Hetzel, Bill.** *The Complete Guide to Software Testing.* 1988.

10. **Davis, A.** *201 Principles of Software Development.* s.l. : MaGraw-Hill, 1995.

11. **Vallejo, Manuel Galán.** *EL MODELO DE EXCELENCIA ORGANIZACIONAL EFQM.* [Online] [Cited: marzo 10, 2008.] http://www.r2h2.us.es/uploads/forpas/cursos_2007/07111_documento_completo_introduccion_efqm.pdf.

12. **Teruel, Alejandro.** El Proceso de Pruebas (Testing). [Online] junio 27, 2001. http://www.ldc.usb.ve/~teruel/ci3715/clases/testingGeneral.html.

13. 2Hsoftware. ***Pruebas y aseguramiento de la calidad***. [Online] 2007. [Cited: febrero 10, 2008.] http://www.2hsoftware.com.mx/serviciosit.html.

14. **Scalone, Fernanda.** Software Quality Management. [Online] Noviembre 1, 2006. [Cited: Febrero 20, 2008.] http://softqm.blogspot.com/2006/11/gestin-de-la-calidad-del-software.html.

15. **Jacobson, Ivar.** *Proceso unificado del desarrollo del software.* 2003.

16. Sopra PROFit. ***Testing de Software***. [Online] 2006. [Cited: febrero 25, 2008.] http://www.profit.es/21verifi.html#menu...

17. **Olivé, Jenisel.** Radio Habana Cuba. [Online] mayo 12, 2005. [Cited: marzo 10, 2008.] http://www.radiohc.cu/espanol/galerias/galeriainformatica2005/mayo05/12may.htm.

18. **Kruchten, P.** *The Rational Unified Process as Introduction. 2 nd Edition.* 2000.

19. **Pfleeger, S.** *Software Engineering.* 1998.

20. **Cardoso, R.** *Pruebas del Software.* 2001.

21. **Sommerville, I.** *Software Engineering.* 2000.

22. **Hernández, Ludisley la Torre.** Propuesta de métrica de perfeccionamiento de gestión de la calidad en el proceso de desarrollo de Software. M*onografías* [Online] [Cited: abril 20, 2008.] http://www.monografias.com/trabajos55/proceso-de-desarrollo-software/proceso-de-desarrollo-software.shtml.

23. **Mañas, José A.** Calidad de las pruebas. [Online] mayo 9, 2003. [Cited: marzo 20, 2008.] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/calidad.htm.

Anexos

Anexo 1

Entrevista realizada a la líder del proyecto Registros y Notarías

Preguntas:

1. ¿Cómo evalúas el proceso de pruebas en el proyecto Registros y Notarías?
2. ¿Cuáles son las mayores dificultades presentadas en la etapa de prueba del proyecto Registros y Notarías?
3. ¿Están bien definidos los roles que interactúan en esta etapa?
4. ¿Existe actualmente en el proyecto alguna herramienta para la automatización de las pruebas?
5. ¿En el proyecto se realizan las pruebas de Caja Blanca?
6. ¿Cuánto crees que influye el proceso de pruebas en la calidad final del producto?

Anexo 2

Procedimientos de prueba de Caja Negra para el caso de uso Registrar Requisiciones en la UEL

Procedimiento: “Eliminar Requisición”

No	Prueba	Acciones a realizar	Resultado esperado
2	Eliminar Requisición.	La Requisición que desee eliminar debe estar en estado de edición, la selecciona y luego presiona el botón Eliminar.	Si la requisición esta en estado de edición y fue seleccionada correctamente el sistema muestra un mensaje de verificación, de que realmente desea eliminarla. Si la respuesta es afirmativa dicha requisición es eliminada.
2.1	Eliminar Requisición.	Seleccione una requisición que no se encuentra en estado de edición.	El sistema debe mostrar un mensaje informándole que la requisición debe encontrarse en estado de

			edición para ser eliminada.

Procedimiento: “Modificar Requisición”

No	Prueba	Acciones a realizar	Resultado esperado
3	Modificar Requisición.	La Requisición que desee modificar debe estar en estado de edición, la selecciona y luego presiona el botón Modificar. El sistema muestra la interfaz Requisición permitiendo modificar los datos deseados, luego presiona el botón Aceptar.	Si la requisición esta en estado de edición y fue seleccionada correctamente el sistema actualiza los datos de dicha requisición.
3.1	Modificar Requisición.	Seleccione una requisición que no se encuentra en estado de edición.	El sistema debe mostrar un mensaje informándole que la requisición debe encontrarse en estado de edición para ser modificada.

Procedimiento: “Autorizar Requisición”

No	Prueba	Acciones a realizar	Resultado esperado
4	Autorizar Requisición.	La Requisición que desee Autorizar debe estar en estado de edición, la selecciona y luego presiona el botón Autorizar.	Si la requisición esta en estado de edición el sistema muestra un mensaje de confirmación. Si la respuesta es afirmativa dicha requisición es autorizada.
4.1	Autorizar Requisición.	Seleccione una requisición que no se encuentra en estado de edición.	El sistema debe mostrar un mensaje informándole que la requisición debe encontrarse en estado de edición para ser

			autorizada.

Procedimiento: “Visualizar Requisición”

No	Prueba	Acciones a realizar	Resultado esperado
5	Visualizar Requisición.	En la interfaz Requisición presione el botón Visualizar.	El sistema muestra un reporte con los datos de la requisición.
5.1	Visualizar Requisición.	Presione el botón visualizar sin haber seleccionado datos.	El sistema no activa el botón Visualizar.

Procedimiento: “Visualizar Registro de Requisiciones”

No	Prueba	Acciones a realizar	Resultado esperado
6	Visualizar Registro de Requisiciones.	En la interfaz Consulta y Registro de Requisiciones seleccione los datos por los que desea realizar la búsqueda y luego presione el botón Buscar. Luego presione el botón Visualizar.	El sistema muestra un reporte con los datos de las requisiciones.
6.1	Visualizar Registro de Requisiciones.	Presione el botón visualizar sin haber realizado la búsqueda o después de realizada la búsqueda sin haber encontrado ninguna requisición.	El sistema no activa el botón Visualizar.

Procedimiento: “Eliminar Bien o Servicio”

No	Prueba	Acciones a realizar	Resultado esperado
7	Eliminar Bien o Servicio.	En la interfaz Requisición seleccione el Bien o Servicio y presione le botón	El sistema muestra un mensaje de verificación, si el mensaje es aceptado el sistema elimina el

		Eliminar.	Bien o Servicio.
7.1	Eliminar Bien o Servicio.	En la interfaz Requisición sin seleccionar un Bien o Servicio presione el botón Eliminar.	El sistema no activa el botón Eliminar.

Procedimiento: “Eliminar Proyecto/Acción Centralizada”

No	Prueba	Acciones a realizar	Resultado esperado
8	Eliminar Proyecto/Acción Centralizada	En la interfaz Requisición seleccione el Proyecto/Acción Centralizada y presione le botón Eliminar.	El sistema muestra un mensaje de verificación, si el mensaje es aceptado el sistema elimina el Proyecto/Acción Centralizada.
8.1	Eliminar Proyecto/Acción Centralizada	En la interfaz Requisición sin seleccionar el Proyecto/Acción Centralizada presione el botón Eliminar.	El sistema no activa el botón Eliminar.

Procedimiento: “Catálogo”

No	Prueba	Acciones a realizar	Resultado esperado
9	Catálogo.	En la interfaz Requisición presione el botón Catálogo mostrándose la interfaz Catálogo de Bienes y Servicios. Presione el botón Buscar si no tiene un criterio de búsqueda. Seleccione el Bien o Servicio deseado y presione el botón Seleccionar.	El sistema actualiza el listado de Bienes y Servicios en la interfaz Requisición.
9.1	Catálogo.	En la interfaz Catálogo de Bienes y Servicios presione el botón	El sistema debe mostrar un mensaje informando que debe

		Seleccionar sin haber seleccionado un Bien o Servicio o sin haber presionado el botón Buscar.	seleccionar un Bien o Servicio.

Anexo 3

Casos de prueba del caso de uso Registrar Requisiciones en la UEL

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: Registrar Requisiciones	EC 1.1: Flujo Básico de Registrar Requisiciones	El sistema permite crear una Requisición	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados 6. El usuario selecciona la opción Crear Requisición. 7. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 8. El usuario selecciona el tipo de requisición y edita la observación de la misma. 9. El usuario selecciona el Bien o Servicio. 10. El usuario edita el dato cantidad. 11. El usuario selecciona Proyecto/Acción Centralizada y la Acción Específica del Bien o Servicio antes seleccionado. 12. El usuario selecciona la opción Aceptar. 13. El sistema Muestra interfaz Registro de

			Requisiciones con lista actualizada.
	EC 1.2: Flujo Alterno Cerrar	El sistema permite cerrar la interfaz sin realizar cambios.	8. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 9. El sistema muestra la interfaz Registro de Requisiciones 10. El usuario selecciona datos. 11. El usuario selecciona la opción Buscar. 12. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 13. El usuario selecciona la opción Cerrar. 14. El sistema no realiza ningún cambio. Cierra la interfaz correspondiente. Terminando así el caso de uso.
	EC 1.3: Flujo Alterno Cancelar	El sistema permite cancelar la interfaz sin realizar cambios.	14. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 15. El sistema muestra la interfaz Registro de Requisiciones 16. El usuario selecciona datos. 17. El usuario selecciona la opción Buscar. 18. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados 19. El usuario selecciona la opción Crear Requisición. 20. El sistema muestra la interfaz Requisición con datos a seleccionar y editar:

			21. El usuario selecciona el tipo de requisición y edita la observación de la misma. 22. El usuario selecciona el Bien o Servicio. 23. El usuario edita el dato cantidad. 24. El usuario selecciona Proyecto/Acción Centralizada y la Acción Específica del Bien o Servicio antes seleccionado. 25. El usuario selecciona la opción Cancelar. 26. El sistema no realiza ningún cambio. Cierra la interfaz correspondiente.
SC 2:Eliminar Requisición	EC 2.1: Flujo Básico Eliminar Requisición	Permite eliminar una Requisición que esté en estado de edición.	8. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 9. El sistema muestra la interfaz Registro de Requisiciones 10. El usuario selecciona datos. 11. El usuario selecciona la opción Buscar. 12. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 13. El usuario selecciona una requisición y ordena Eliminar Requisición. 14. El sistema comprueba que la Requisición se encuentre en estado de edición, si es así se elimina.
	EC 2.2: Flujo Alterno Eliminar Requisición	El sistema no permite eliminar una Requisición que no esté en	8. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 9. El sistema muestra la interfaz Registro de

		estado de edición.	Requisiciones 10. El usuario selecciona datos. 11. El usuario selecciona la opción Buscar. 12. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 13. El usuario selecciona una requisición y ordena Eliminar Requisición. 14. El sistema comprueba que la Requisición se encuentre en estado de edición, si es así no la elimina.
SC 3: Modificar Requisición	EC 3.1: Flujo Básico Modificar Requisición	El sistema permite modificar una Requisición.	14. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 15. El sistema muestra la interfaz Registro de Requisiciones 16. El usuario selecciona datos. 17. El usuario selecciona la opción Buscar. 18. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 19. El usuario selecciona la opción Modificar Requisición 20. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 21. El usuario selecciona el tipo de requisición y edita la observación de la misma.

			22. El usuario selecciona el Bien o Servicio. 23. El usuario edita el dato cantidad. 24. El usuario selecciona Proyecto/Acción Centralizada y la Acción Específica del Bien o Servicio antes seleccionado. 25. El usuario selecciona la opción Aceptar. 26. El sistema Muestra interfaz Registro de Requisiciones con lista actualizada.
	EC 3.2: Flujo Alterno Modificar Requisición	El sistema permite cancelar la modificación realizada	14. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 15. El sistema muestra la interfaz Registro de Requisiciones 16. El usuario selecciona datos. 17. El usuario selecciona la opción Buscar. 18. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 19. El usuario selecciona la opción Modificar Requisición 20. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 21. El usuario selecciona el tipo de requisición y edita la observación de la misma. 22. El usuario selecciona el Bien o Servicio. 23. El usuario edita el dato cantidad. 24. El usuario selecciona Proyecto/Acción

			Centralizada y la Acción Específica del Bien o Servicio antes seleccionado. 25. El usuario selecciona la opción Cancelar. 26. El sistema no realiza ningún cambio. Cierra la interfaz correspondiente.
SC 4: Autorizar Requisición	EC 4.1: Flujo Básico Autorizar Requisición	Permite autorizar la requisición si está en estado de edición	11. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 12. El sistema muestra la interfaz Registro de Requisiciones 13. El usuario selecciona datos. 14. El usuario selecciona la opción Buscar. 15. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 16. El usuario selecciona la opción Autorizar. 17. El sistema verifica que la requisición esté en estado de edición. 18. El sistema muestra interfaz Autorización de la Requisición donde se confirma la autorización de la requisición. 19. El usuario selecciona la opción Aceptar. 20. El sistema muestra interfaz Registro de Requisiciones con cambios en el estado de las requisiciones.

	EC 4.2: Flujo alterno Cancelar	Permite cancelar la operación	11. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 12. El sistema muestra la interfaz Registro de Requisiciones 13. El usuario selecciona datos. 14. El usuario selecciona la opción Buscar. 15. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 16. El usuario selecciona la opción Autorizar. 17. El sistema verifica que la requisición esté en estado de edición. 18. El sistema muestra interfaz Autorización de la Requisición donde se confirma la autorización de la requisición. 19. El usuario selecciona la opción Cancelar. 20. El sistema no realiza ningún cambio. Cierra la interfaz correspondiente.
SC 5: Visualizar Requisición	EC 5.1: Flujo Básico Visualizar Requisición	Permite visualizar	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados 6. El usuario selecciona la opción Crear

			Requisición. 7. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 8. El usuario selecciona el tipo de requisición y edita la observación de la misma. 9. El usuario selecciona el Bien o Servicio. 10. El usuario edita el dato cantidad. 11. El usuario selecciona Proyecto/Acción Centralizada y la Acción Específica del Bien o Servicio antes seleccionado. 12. El usuario selecciona la opción Visualizar Requisición 13. El sistema visualiza el Reporte de Requisiciones.
SC 6: Visualizar Registro de Requisiciones	EC 6.1: Flujo Básico Visualizar Requisición	Permite visualizar el Registro de Requisiciones	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados. 6. El usuario selecciona la opción Visualizar Registro de Requisiciones 7. El sistema visualiza el Reporte de Registro de Requisiciones.

SC 7: Eliminar Bien o Servicio	EC 7.1: Flujo Básico Eliminar Bien o Servicio	Permite eliminar un Bien o Servicio	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados 6. El usuario selecciona la opción Crear Requisición. 7. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 8. El usuario selecciona el tipo de requisición y edita la observación de la misma. 9. El usuario selecciona el Bien o Servicio. 10. El usuario selecciona la opción Eliminar Bien o Servicio. 11. El sistema elimina el Bien o Servicio para que no sea incluido en la requisición.
SC 8: Eliminar Proyecto / Acción Centralizada	EC 8.1: Flujo Básico Eliminar	Permite eliminar un Proyecto / Acción Centralizada	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones

			asociados a los datos antes seleccionados 6. El usuario selecciona la opción Crear Requisición. 7. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 8. El usuario selecciona el tipo de requisición y edita la observación de la misma. 9. El usuario selecciona el Bien o Servicio. 10. El usuario edita el dato cantidad. 11. El usuario selecciona Proyecto/Acción Centralizada y la Acción Específica del Bien o Servicio antes seleccionado. 12. El usuario selecciona la opción Eliminar Proyecto / Acción Centralizada. 13. El sistema elimina el Proyecto / Acción Centralizada para que no sea incluido en la requisición.
SC 9: Catálogo	EC 9.1: Flujo Básico Catálogo	Permite realizar operaciones en el catálogo de bienes y servicios	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar. 5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados 6. El usuario selecciona la opción Crear

			Requisición. 7. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 8. El usuario selecciona el tipo de requisición y edita la observación de la misma. 9. El usuario selecciona opción Catálogo. 10. El sistema muestra la interfaz Catálogo de Bienes y Servicios con dato a editar para realizar una búsqueda de un bien o servicio. 11. El usuario edita los datos. 12. El usuario selecciona la opción Buscar. 13. El sistema señala el Bien o Servicio encontrado en caso de que exista con su código y denominación. 14. El usuario selecciona uno o varios bienes o servicios de la lista. 15. El usuario selecciona la opción Seleccionar. 16. El sistema muestra la interfaz Requisición con la lista actualizada.
	EC 9.1: Flujo Alterno Cancelar	Permite cancelar la operación	1. El usuario ordena Consultar y Registrar Requisiciones en la UEL. 2. El sistema muestra la interfaz Registro de Requisiciones 3. El usuario selecciona datos. 4. El usuario selecciona la opción Buscar.

			5. El sistema muestra lista de Requisiciones asociados a los datos antes seleccionados 6. El usuario selecciona la opción Crear Requisición. 7. El sistema muestra la interfaz Requisición con datos a seleccionar y editar: 8. El usuario selecciona el tipo de requisición y edita la observación de la misma. 9. El usuario selecciona opción Catálogo. 10. El sistema muestra la interfaz Catálogo de Bienes y Servicios con dato a editar para realizar una búsqueda de un bien o servicio. 11. El usuario edita los datos. 12. El usuario selecciona la opción Buscar. 13. El sistema señala el Bien o Servicio encontrado en caso de que exista con su código y denominación. 14. El usuario selecciona uno o varios bienes o servicios de la lista. 15. El usuario selecciona la opción Cancelar. 16. El sistema no realiza ningún cambio. Cierra la interfaz correspondiente.

Anexo 4

Definición de las métricas para mediciones

1. Densidad de defectos.

$$DD = \frac{DT}{TP}$$

DT: cantidad total de defectos encontrados en el producto.

TP: tamaño del producto, puede ser estimado en líneas de código (en miles) o en puntos de función.

Da una medida de la proporción de defectos del producto con respecto a su tamaño. De la forma en que está definida debe ser evaluada al finalizar el producto y puede ser aprovechada como experiencia para proyectos futuros, no obstante en las etapas tempranas de desarrollo del proyecto esta métrica puede ser utilizada considerando la estimación preliminar de líneas de código o puntos de función que haya sido considerada en la planificación del proyecto.

2. Cantidad total de líneas revisadas.

$$TLCR = \sum_{k=1}^{n} LCR_k$$

LCRk: cantidad de líneas de código revisadas en la revisión k.

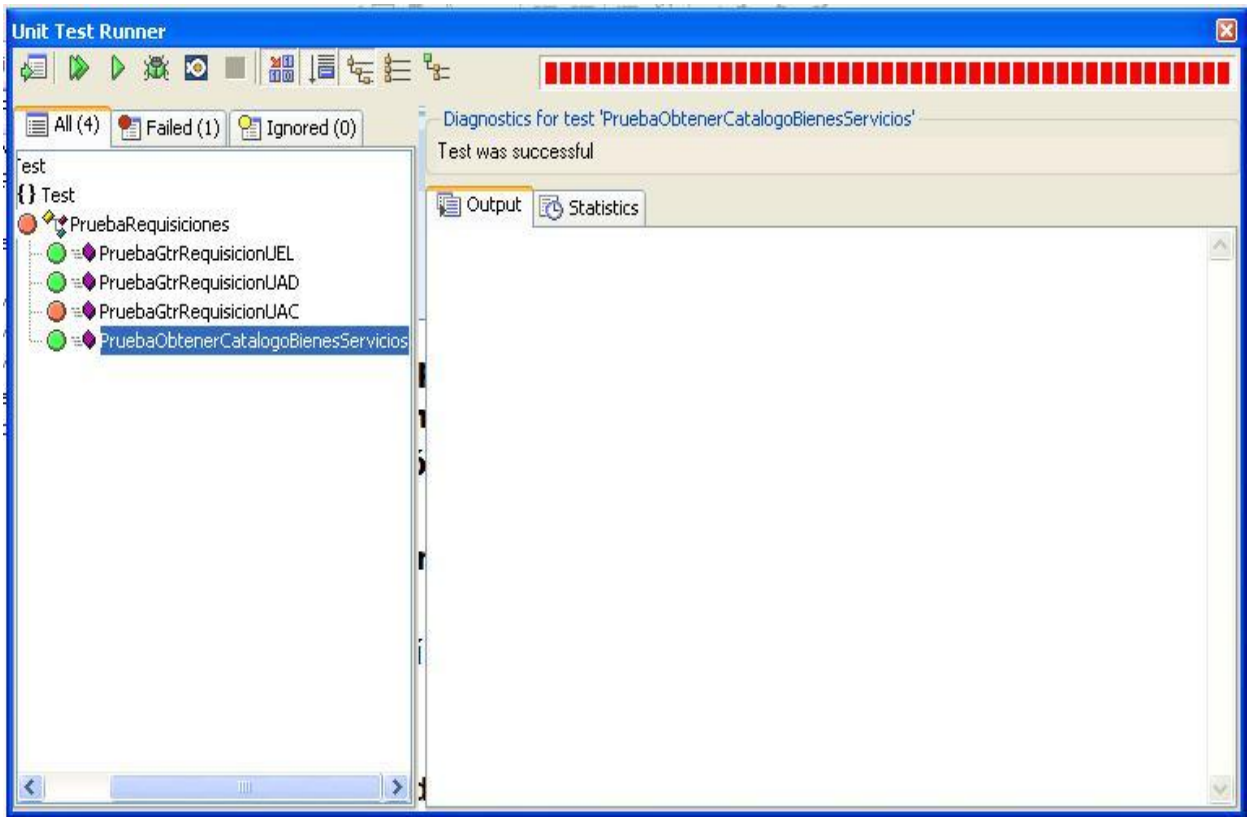
n: número total de revisiones.

3. Promedio de líneas revisadas en cada revisión.

$$PLCR = TLCR/n$$

Anexo 5

Pruebas de Caja Blanca realizadas en NUnit.



Anexo 6

Clasificación de los defectos

Según la prioridad

Prioridad	***Significado***
Urgente	Se definen para aquellos defectos que no permiten la ejecución de otras funcionalidades del sistema.
Alto	Se definen para aquellos defectos que causan que el sistema se comporte de forma incorrecta.
Normal	Se definen para aquellos defectos que afectan la estética del sistema.

Según la severidad

Severidad	***Significado***
Crítico	El defecto causa un fallo en el sistema, subsistema, o módulo dentro del sistema.
Alto	El defecto no causa un fallo, pero causa que el sistema se comporte de forma incorrecta, incompleta o con resultados inconsistentes, o comprometa la usabilidad del sistema.
Bajo	El defecto es estético o es el resultado de una no conformidad con algún estándar.