UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

UCI

FACULTAD 15

Trabajo de Diploma

Presentado para optar por el título de Ingeniero en Ciencias Informáticas.

Título: Estrategia de Pruebas para el Proyecto Sistema Automatizado para la Gestión Bancaria (SAGEB).

Autores:

Tatiana Rosabal González.

Yarida Padrón Benítez.

Tutores:

Ing. Yadian Guillermo Pérez Betancourt.

Ing. Liset González Polanco.

Ciudad de la Habana, Cuba. Junio 2010. “Año 52 de la Revolución”.

DECLARACIÓN DE AUTORÍA

Declaramos que somos los autores de este trabajo y autorizamos a la Facultad 15 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los _10___días del mes de __Junio_____ del año 2010.

AUTOR

Tatiana Rosabal González.

AUTOR

Yarida Padrón Benítez.

TUTOR

ING. Liset González Polanco.

TUTOR

ING. Yadian Pérez Betancourt.

Datos de Contacto

Ing. Liset González Polanco Graduado de Ingeniero en Ciencias Informáticas en el 2009. Título de Oro. Actualmente imparte clases de Contabilidad y Finanzas y Administración de Empresas en la Facultad 4, integra el proyecto Modernización del Sistema Bancario Cubano y se desempeña como Jefe de Calidad del mismo.

Ing. Yadian Guillermo Pérez Betancourt Graduado de Ingeniero en Ciencias Informáticas en el 2009. Título de Oro. Premio Mella. Cumplió misión en Venezuela. Actualmente imparte clases de Programación II y Programación IV en la Facultad 4, integra el proyecto Modernización del Sistema Bancario Cubano y se desempeña como Gestor de la Configuración en el mismo.

Resumen

Resulta un poco común y en forma de eslogan escuchar que los productos de software deben tener “Calidad”, es una palabra corta pero que significa mucho en la industria de software y lamentablemente con solo escribirla no se logra.

La importancia de realizar una captura de requisitos acertada, una descripción y un seguimiento adecuado dan en gran medida que el eslogan “calidad “se cumpla, porque tienen la propiedad de que si no se corrigen a tiempo influirán directamente en las etapas posteriores del desarrollo de software, trayendo como consecuencia la obtención de un producto distinto al que necesita el usuario.

Una buena práctica para la obtención de un producto de alta calidad es la ejecución de pruebas que permitan una evaluación del trabajo en el proyecto, donde sus resultados muestren las deficiencias del software así como las posibles mejoras.

Autores prestigiosos como Pressman 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” y en la metodología RUP proporciona esa guía.

Crear un sistema que cumpla con los requisitos de los clientes se ha vuelto la gran dificultad que viven las empresas productoras de software a la medida y dificultad que afecta también a la Universidad de las Ciencias Informáticas (UCI). Primera Universidad surgida al calor de la batalla de ideas y que tiene entre sus misiones contribuir a la informatización de las principales organizaciones y entidades del país.

La facultad 15 es la encargada en la UCI del desarrollo del proyecto Sistema Automatizado para la Gestión Bancaria (SAGEB), con el objetivo de mejorar los servicios del Banco Nacional de Cuba. Tras un minucioso estudio de la situación y de las posibles soluciones, se concluyó que la calidad del producto dependerá del seguimiento de una Estrategia de Pruebas adecuada a las características del proyecto. En este documento se plantea dicha estrategia, la cual está fragmentada en cuatros niveles fundamentales (unidad, integración, sistema y aceptación) dentro de los cuales se proponen tipos de pruebas, enfocados a las estructuras priorizables del producto.

Índice de Contenido

Índice de Figuras.

Índice de Tablas.

Introducción

La tecnología acelera su desarrollo, permitiendo a los seres humanos una vida más fácil y próspera. Las nuevas tecnologías constituyen una herramienta básica para el desarrollo de cada una de las esferas de la sociedad. La información se ha convertido en un importante factor de competitividad, lo que obliga a la utilización de nuevas estrategias y herramientas de una forma cada vez más eficiente.

El aseguramiento de la calidad en el software es una operación que ha surgido como consecuencia de la insaciable demanda de sistemas de software de todos los procesos que se desarrollan en la actualidad, desde los más simples hasta los más complejos.

Todas las tareas están conectadas a una computadora, la cual hace más rápido, fácil y organizado el trabajo, por lo que se hace ineludible la elaboración de productos de software que satisfagan cada una de las necesidades requeridas, con el uso preciso de los recursos necesarios. Una actividad fundamental en el proceso de desarrollo de software es la relativa a Prueba, las cuales generalmente no se implementan de forma sistemática y organizada, es esta la causa del fracaso de muchos proyectos a nivel mundial, donde se desperdicia trabajo, tiempo y recursos materiales sin obtener a cambio un producto utilizable.

El único modo de lograr un trabajo realmente eficiente es analizar cada paso, prever cada consecuencia y rectificar errores. Una de las principales acciones que se han realizado en pos del desarrollo de la informática en Cuba, es la creación de la Universidad de la Ciencias Informáticas (UCI), la cual está equipada para la producción de todo tipo de software, respaldados por un Departamento Central de Calidad encargado de garantizar que los productos que se realizan estén acorde con las solicitudes y expectativas de los clientes. La UCI capacita a jóvenes con el objetivo de elevar a Cuba a una mejor posición en el mercado informático, tomando como premisa que el producto de software que se produzca, antes de ser liberado, debe pasar por una ardua fase de prueba que garantice la calidad del software. Dicha universidad cuenta con diferentes tipos de clientes desde entidades cubanas hasta empresas extranjeras los cuales confían en la calidad de los productos que solicitan. Una de las entidades que ha solicitado los servicios de la universidad, es el Banco Nacional de Cuba, entidad que presenta

ineficiencias por la utilización del Sistema Automatizado para la Banca Internacional de Comercio (SABIC). La plataforma tecnológica sobre la que está basado este sistema contiene un conjunto de limitaciones entre las que se encuentra la mono tarea, que está directamente asociada a la utilización del MS-DOS como sistema operativo, por lo cual no puede llevar a cabo de una manera efectiva y menos compleja la toma de decisiones, la obtención de los reportes y el resto de sus operaciones. Además resulta extremadamente engorroso, trabajar sobre varias plataformas.

Teniendo en cuenta estas situaciones la Universidad le ha dado la responsabilidad a la Facultad 15 de la creación del proyecto Sistema Automatizado para la Gestión Bancaria (SAGEB). Dicho proyecto requiere obtener un producto de calidad, por lo que se plantea como **Problema a resolver**: ¿Cómo se deben realizar las pruebas durante el ciclo de vida del proyecto SAGEB para asegurar la calidad del producto? El **objeto de estudio** del presente trabajo se enfocará en los estándares y guías existentes referentes a la realización de las pruebas de software. Y el **Campo de Acción** el flujo de trabajo de pruebas dentro del proceso de desarrollo de software del proyecto SAGEB.

Para responder al problema anteriormente planteado, los autores se trazaron como **Objetivo General** Proponer una estrategia para la realización de las pruebas de software en el proyecto SAGEB, para darle cumplimento a dicho objetivo se trazan los siguientes **Objetivos específicos.**

1. Caracterizar estándares, normas y modelos para la realización de pruebas de software.
2. Evaluar Planes de Pruebas de proyectos similares, herramientas y recursos para la realización de las pruebas en el proyecto SAGEB.
3. Definir cómo serán evaluados los riesgos que puedan surgir en el proceso de pruebas.
4. Definir la estrategia para la realización de las pruebas de software en el proyecto SAGEB.
5. Elaborar un Plan de Pruebas para el Proyecto SAGEB.

Con el cumplimiento satisfactorio de los objetivos trazados se espera como **Posible resultado** obtener una estrategia de pruebas para el proyecto SAGEB, la cual garantice un producto de alta calidad.

La investigación está estructurada por: Introducción, tres Capítulos, Conclusiones, Recomendaciones, Referencia bibliográfica, Bibliografía, Glosario de términos y Anexos.

Capítulo 1 Fundamentación teórica: Contiene conceptos relacionados con las pruebas, como artefactos de prueba, estrategia de pruebas, metodología, tipos y niveles de prueba. Se hace mención de herramientas para la automatización de las pruebas y se realiza una breve caracterización de normas y estándares para la realización de pruebas. Se analizan estrategias ya definidas en la universidad comparándolas con las necesidades del proyecto.

Capítulo 2 Propuesta de estrategia: Contiene las descripción de la estrategia que se plantea especificando el entono de prueba, el grupo de trabajo, los niveles y tipos de prueba que se realizarán con la correspondiente descripción de cada una, además de los riesgos que pueden presentarse al implantar la estrategia y la posible solución de los mismos.

Capítulo 3 Resultados obtenidos: Contiene los resultados parciales que se han obtenido durante la implantación de la estrategia en el proyecto SAGEB.

Capítulo 1: Fundamentación Teórica.

1.1 Introducción

Antes de elaborar una de estrategia es necesario tener claridad con respecto a algunas terminologías relacionadas con el proceso de pruebas y la descripción de sus herramientas.

Estos serían los puntos de partida para con la práctica, asegurar que un software cuente con la excelencia requerida. En este capítulo se aborda todo lo relacionado con estas temáticas, aportando un resumen de concepciones que será de gran utilidad en la comprensión de las mismas y un análisis relacionado con las diferentes estrategias propuestas en la universidad con el fin de lograr de forma productiva una nueva estrategia que se adapte a las necesidades de este proyecto SAGEB.

A continuación se definen los conceptos fundamentales referentes a calidad.

1.2 Calidad del software.

Partiendo de que la calidad es un conjunto de propiedades inherentes a un objeto que le confieren capacidad para satisfacer necesidades implícitas o explícitas [5], se podría definir básicamente como un software de calidad a aquel capaz de satisfacer los requerimientos predefinidos por los implicados. Según Roger S. Pressman la calidad de software es:

“Concordancia con los requisitos funcionales y de rendimiento, explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente”. [1]

La evaluación de la calidad puede ser imprecisa según los gustos y preferencias de los usuarios finales, pues lo que para algunos puede ser deficiente para otros podría resultar lo ideal, pero a pesar de estas ambigüedades se puede decir que los usuarios consideran que un software tiene calidad si cumple o excede las expectativas en cuanto a:

- Funcionalidad: que cumpla los requerimientos trazados.
- Usabilidad: que sea de fácil manejo y de buena estética.
- Confiabilidad: que sea capaz de almacenar información segura al grado que se requiera.
- Performance: capacidad de rendimiento.
- Soportabilidad: que asegure el funcionamiento en diferentes configuraciones de hardware y software.

Es importante destacar que realmente son muchos los factores que deben tomarse en cuenta para que un producto de software tenga calidad.

1.2.1 Niveles de la calidad

La calidad del software puede gestionarse a diferentes niveles, esto proporciona la gestión de la calidad desde las células hasta el todo en el proceso de desarrollo de software. Existen tres niveles de la calidad que se relacionan a continuación:

- ✓ A nivel de producto: Cuando se centran en el proceso de desarrollo de software y se realiza un seguimiento durante todo el proceso, con el objetivo detectar en el trayecto de la producción la existencia de defectos en el producto.
- ✓ A nivel de proceso: Cuando se centran en gestionar todas las áreas de procesos operativos de una organización, mediante la implantación de una metodología. Así se consigue tener mayor información de los procesos de modo que puedan controlarse y mejorarse, un proceso implica un esfuerzo permanente y repetitivo, por lo que con la gestión de su calidad se produce una mejora continua que valide los productos y servicios asociados.

- ✓ A nivel de proyecto: Cuando se centran en controlar todas las fases y áreas de gestión de proyecto, es decir en la definición, planificación, seguimiento y evaluación del proyecto, donde juegan un papel fundamental los individuos y líderes que gestionan las funciones básicas, implantando metodologías y mejores prácticas que aseguren la correcta gestión de las mismas.

1.3 Prueba.

Una prueba de software es un proceso en el cual se ejecuta un sistema o uno de sus componentes en circunstancias previamente especificadas, los cuales se registran, se analizan y se realiza una evaluación a partir de los resultados obtenidos, con el objetivo final de limar asperezas en el sistema de producción y prevenir así la acumulación de trabajo deficiente. El objetivo de la prueba: “es diseñar pruebas que saquen a la luz diferentes clases de errores con la menor cantidad de tiempo y espacio” [1].

1.3.1 Objetivo de las pruebas.

Glen Myers establece una serie de reglas que se pueden tomar como objetivos de la prueba:

- ✓ La prueba es un proceso de ejecución de un programa con la intención de descubrir errores.
- ✓ 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.

Básicamente las pruebas se realizan con el fin de garantizar un software con calidad, mediantes la detección de errores, por lo que sería una prueba exitosa si es capaz de sacar los errores con el empleo de la menor cantidad de tiempo y esfuerzo.

“La prueba no puede asegurar la ausencia de defectos; solo puede demostrar que existen defectos en el software”. [1]

1.3.2 Atributos que definen una buena prueba

Una buena prueba es aquella que tiene altas posibilidades de detectar un error.

De acuerdo con lo citado en Ingeniería de Software un enfoque práctico, donde se plantea que según Kaner, Falk y Nguyen se sugieren algunas características que debe tener una *buena prueba*:

- ✓ *Una buena prueba tiene una alta probabilidad de encontrar un error.* El ingeniero de software debe tener un amplio conocimiento del software a construir para poder diseñar buenos casos de prueba que sean capaces de detectar el mayor número de defectos.
- ✓ *Una buena prueba no debe ser redundante.* Uno de los objetivos de las pruebas es *«encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles»*, se debe buscar diseñar el menor número de casos de prueba que permitan probar adecuadamente el software y de esta forma se optimizan los recursos. Cada una de las pruebas debe tener un objetivo diferente.
- ✓ *Una buena prueba debería ser la mejor de la cosecha.* La abreviación del tiempo y los recursos sugiere que sean aplicadas las pruebas que tengan más probabilidades de detectar errores.
- ✓ *Una buena prueba no debería ser ni demasiado sencilla ni demasiado compleja.* [1]

1.3.3 Principios de las pruebas

Pressman en su libro menciona algunos de los principios básicos sugeridos por Davis que guían las pruebas del software:

- ✓ *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.* La planificación de las pruebas puede empezar tan pronto como esté completo el modelo de requisitos y la definición detallada de los casos de pruebas
- ✓ *Las pruebas deberían empezar por «lo pequeño» y progresar hacia «lo grande».* Es decir comenzar por los módulos para luego irse integrando con el resto, desde las células hasta el todo.
- ✓ *No son posibles las pruebas exhaustivas.* Se deben tomar los caminos lógicos más importantes para la realización de las pruebas y no tratar de abarcar todos los caminos que guíen a una prueba.
- ✓ *Para ser más eficaces, las pruebas deberían ser realizadas por un equipo independiente.* Los ingenieros que crearon el sistema no son los más indicados para realizar las pruebas pues podrían omitir pruebas que consideren innecesarias y que no lo sean, es sugerente que las hagan personas especializadas que no hayan participado directamente en la construcción. [1]

1.3.4 Niveles de Prueba

Una prueba puede ser aplicada en diferentes escenarios o niveles de trabajo. Dentro de estos niveles existen diferentes tipos de pruebas, estos niveles se agrupan en dos categorías atendiendo a quién ejecuta las pruebas, pueden ser independientes o de desarrollador.

- **Prueba de Desarrollador**. Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas pruebas han sido consideradas solo para la prueba de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración.

➢ **Nivel de integración:** Es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación. Estas pruebas descubren errores o incompletitud en las especificaciones de las interfaces de los paquetes. Esta prueba debe ser responsabilidad de desarrolladores y de independientes, sin solaparse las pruebas.

Es el proceso de combinar y probar múltiples componentes juntos. El objetivo es tomar los componentes probados en unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.

Se llama integración incremental cuando el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir, es más probable que se pueda probar completamente las interfaces y se puede aplicar un enfoque de prueba sistemática. Hay dos estrategias de integración: incremental ascendente e Incremental descendente.

- ✓ ***Integración Descendente (Top-Down):***

Se integran los módulos moviéndose hacia abajo por la jerarquía de control. Comenzando por el módulo principal, los módulos subordinados se van incorporando a la estructura, bien en forma *primero en profundidad,* que integra todos los módulos de un camino de control principal de la estructura, o *primero*

en anchura, que incorpora todos los módulos directamente subordinados a cada nivel, moviéndose por la estructura de forma horizontal.

- ✓ ***Integración Ascendente*** **(Bottom-Up):**

Empieza la construcción y la prueba con los módulos de los niveles más bajos de la estructura del programa. Dado que los módulos se integran de abajo hacia arriba, el proceso requerido de los módulos subordinados a un nivel dado siempre está disponible y se elimina la necesidad de resguardos.

En el caso de integrar varios módulos y se encuentra un error en el momento de integrarlos, se tiene que hacer una *Prueba de Regresión*.

Prueba de Regresión**:** Cada vez que se añade un nuevo módulo como parte de una prueba de integración, el software cambia.

Estos cambios pueden causar problemas con funciones que antes trabajaban perfectamente. La prueba de regresión es la actividad que ayuda a asegurar que los cambios no introducen un comportamiento no deseado o errores adicionales. El conjunto de pruebas de regresión contiene tres clases diferentes de casos de prueba:

• Una muestra representativa de pruebas que ejercite todas las funciones del software.

• Pruebas adicionales que se centran en las funciones del software que se van a ver probablemente afectadas por el cambio.

• Pruebas que se centran en los componentes del software que ha cambiado.

No es práctico ni eficiente volver a ejecutar cada prueba de cada función del programa después de un cambio.

- **Prueba independiente**: Es la prueba que es diseñada e implementada por alguien independiente del grupo de desarrolladores. El objetivo de estas pruebas es proporcionar una perspectiva diferente y en un ambiente más rico que los desarrolladores. Una vista de la prueba independiente es la prueba independiente de los stakeholder, que son pruebas basadas en las necesidades y preocupaciones de los stakeholders. Esta categoría engloba los niveles de Unidad, Sistema y Aceptación.

- **Nivel de unidad**: Es la prueba enfocada a los elementos testeables más pequeño del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del componente. Si los datos no entran correctamente, todas las demás pruebas no tienen sentido. Las pruebas que se realizan en este nivel pueden ser mediante la utilización de dos métodos: Caja blanca o caja negra. Es importante aclarar que estos métodos se utilizan en cualquiera de los niveles definidos.

  - **La prueba de caja negra** son las pruebas que se realizan sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funcionalidades del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Son conocidas por pruebas de caja opaca, o pruebas de entrada/salida, o pruebas inducidas por los datos.

    Se verifican las especificaciones funcionales y no consideran la estructura interna del programa. Son hechas sin el conocimiento interno del producto. No validan funciones ocultas por tanto los errores asociados a ellas no serán encontrados.

    En otras palabras, la prueba de la caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea los casos de prueba pretenden:

    1. Demostrar las funciones del software son operativas.
    2. Que las entradas se aceptan de la forma adecuada y que se produce el resultado correcto.

3. Así como la integrada de la información externa (por ejemplo archivos de datos) se mantiene.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

✓ **Técnica de la Partición de Equivalencia:** esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

✓ **Técnica del Análisis de Valores Límites:** esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

✓ **Técnica de Grafos de Causa-Efecto**: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

- **Las pruebas de la caja blanca** son las que comprueban los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado. Son conocidas por pruebas estructurales o pruebas de caja transparente.

Métodos para aplicar la Prueba de caja blanca son:

✓ **La prueba del camino básico*:*** Esta prueba 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.

✓ Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

✓ **La prueba de condición*:*** Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

✓ **La prueba de flujo de datos**: Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

✓ **La prueba de bucles:** Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

- **Nivel de Sistema**: Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. Dentro de este nivel existen varios tipos de pruebas clasificados según los requisitos del sistema: funcionales o no funcionales

  - ***Funcionales***

Enfocadas a los requisitos funcionales del software. Comprueban el correcto comportamiento de las acciones que debe realizar el sistema.

- ✓ **Funcionalidad:** Estas pruebas se caracterizan por la validación de las funciones, métodos, servicios y caso de uso.

  - ***No funcionales***

- ✓ **Pruebas de Resistencia**: Ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales. Por ejemplo: ejecutar casos de prueba que requieran el máximo de memoria o de otros recursos.

- ✓ **Pruebas de Volumen:** Someten a la aplicación a volúmenes pesados de datos y tienen como objetivo demostrar que el programa no puede manejar el volumen de datos. Puesto que las pruebas de volumen pueden requerir una cantidad de recursos considerables, en términos de cantidad de máquinas y de tiempo por parte de las personas, no se puede ir demasiado lejos. Sin embargo, todo programa debe ser expuesto a pruebas de volumen

- ✓ **Pruebas de Rendimiento o Carga:** Verificar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado y la capacidad del sistema para manejar volúmenes de datos extremos de acuerdo al tiempo de respuesta establecido para el sistema.

- ✓ **Las pruebas de seguridad**: intentan verificar que los mecanismos de protección incorporados en el sistema lo protegerán, de hecho, de accesos impropios. Durante la prueba de seguridad, el

responsable de la prueba desempeña el papel de un individuo que desea entrar en el sistema y debe intentar conseguir las claves de acceso por cualquier medio, puede atacar al sistema con software diseñado para romper cualquier defensa.

- ✓ **Usabilidad:** Estas prueba se enfocan en trabajar con los factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento.

- ✓ **Integridad:** Estas pruebas se enfocan en trabajar con la valoración de la robustez (resistencia a fallos).

- ✓ **Estructura:** Estas pruebas se enfocan en trabajar con la adherencia a su diseño y formación. Este tipo de prueba es hecho a las aplicaciones Web asegurando que todos los enlaces están conectados, el contenido deseado es mostrado y no hay contenido huérfano.

- ✓ **Stress:** Estas pruebas se enfocan en evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible).

- ✓ **Contención:** Estas prueba se enfocan en la validación de las habilidades del elemento a probar para manejar aceptablemente la demanda de múltiples actores sobre un mismo recurso (registro de recursos, memoria, etc).

- ➢ **Nivel de aceptación**: Prueba de aceptación del usuario es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Prueba Alfa: Esta prueba es llevada a cabo por un cliente en el lugar de desarrollo. El software es usado de forma natural con el desarrollador como observador del usuario. Las pruebas alfas se llevan a cabo en

un entorno controlado. Para que estas pruebas sean válidas primero debe crearse un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente, una vez logrado esto se procede a realizar las pruebas y a documentar los resultados.

Prueba beta: Esta prueba es llevada a cabo por los usuarios finales del software, a diferencia de la prueba alfa el desarrollador no está presente. Esto convierte a la prueba beta en una aplicación en vivo del software, en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador.

1.4 Estrategia de Prueba.

Como premisa para obtener un producto de alta calidad es necesaria la realización de pruebas y esas pruebas deben estar organizadas para garantizar su efectividad de ahí la necesidad de realizar una estrategia de prueba, dicha estrategia debe abarcar planificación, el diseño de casos de prueba, la ejecución y los resultados, también se deben tener en cuenta los recursos que se van a emplear tanto humanos como materiales. La estrategia de prueba integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba. Incluye los niveles de prueba y el tipo de prueba a ser ejecutada. La estrategia de pruebas debe incluir pruebas de bajo nivel que verifiquen que todos los pequeños segmentos de código fuente se han implementado correctamente, así como pruebas de alto nivel que validen las principales funciones del sistema frente a los requisitos del cliente. Una estrategia debe proporcionar una guía al profesional y un conjunto de hitos para el jefe de proyecto. Debido a que los pasos de la estrategia de prueba se dan a la vez cuando aumenta la presión de los plazos fijados, se debe poder medir el progreso y los problemas deben aparecer lo antes posible. [1]

Dichas estrategias contienen diferentes niveles de pruebas:

- Pruebas de unidades.
- Pruebas de integración.

- Pruebas de sistema.
- Pruebas de aceptación.

En cada nivel de pruebas se analizará el plan de pruebas para los tipos de pruebas a aplicar. También se analizarán los recursos recomendados para hacer las pruebas, la composición de personal para el proyecto y los hitos del proceso. Se especificarán los casos de prueba, los resultados obtenidos así como un resumen de las pruebas, la evaluación y los costos.
Básicamente busca dejar claros los siguientes aspectos que atañen al proceso de pruebas de un desarrollo de software.

- ¿Qué metodología de pruebas se va a usar?
- ¿Cómo serán evaluados los riesgos del software?
- ¿Qué técnicas específicas de pruebas serán usadas para probar el software?
- ¿Cuáles serán los criterios para decidir si avanzar o quedarse en un nivel de pruebas?
- ¿Qué herramientas y recursos serán necesarios?
- ¿Cuál será el entorno de pruebas?

En cuanto al equipo de pruebas se debe tener en cuenta cantidad de personal necesario y las habilidades deben tener; además del entrenamiento adicional para dicho personal.

1.5 Metodología de desarrollo de software: Proceso unificado de desarrollo de software.

Es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software [2]. Sin embargo, el proceso unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

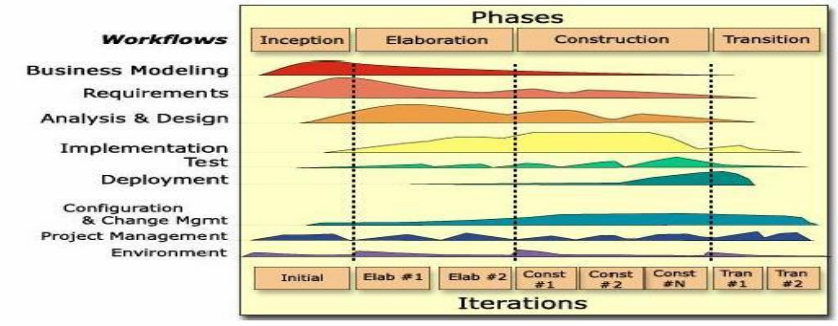

Figura 1: RUP en dos dimensiones.

Tabla 1: Procesos de RUP.

Como RUP es un proceso, en su modelación define como sus principales elementos: Trabajadores (quién)	Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
Actividades (cómo)	Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
Artefactos (qué)	Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
Flujo de actividades (Cuándo)	Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

1.5.1 Descripción de los artefactos de pruebas definidos por la metodología RUP.

1.5.1.1 Caso de prueba

El diseño de casos de prueba es uno de los pasos más importantes al realizar una estrategia de pruebas, ya que estos constituyen la fuente para evaluar los resultados del software. Especifican una forma de probar el sistema, incluyendo la entrada o resultado con que se va a probar y las condiciones bajo las que ha de probarse. Una de las principales características que deben presentar los casos de prueba es su carácter abarcador, es decir, deben ser completos y estar adaptados al producto, pues deben ofrecernos la posibilidad de encontrar la mayor cantidad de errores posibles en un tiempo y costo no muy elevados. [2]. La plantilla del diseño de casos de pruebas se encuentra en los anexos de este documento.

1.5.2.3 Procedimiento de Prueba

Un procedimiento de prueba especifica cómo realizar uno o varios casos de pruebas ó partes de éstos. Un procedimiento de prueba puede ser una instrucción para un individuo sobre cómo realizar un caso de prueba manualmente, ó una especificación de cómo interactuar manualmente con una herramienta de automatización de pruebas, para crear componentes ejecutables de prueba.

1.5.2.5 Plan de prueba

La construcción de un buen Plan de Pruebas es el principal factor de éxito para la puesta en práctica de una estrategia de pruebas que permita entregar un software de mejor nivel. Es el artefacto que permite trazar el tipo de prueba que se le va a aplicar al producto, cuyo propósito es dejar de forma explícita el alcance, el enfoque, los recursos requeridos, el calendario y los responsables del proceso de pruebas. Durante el desarrollo del software se diseña el plan de prueba con el objetivo de asegurar que todos los requisitos tanto funcionales como de rendimiento se satisfagan. [2]. La plantilla de plan de pruebas se encuentra en los anexos de este documento.

1.5.2.6 Resumen de prueba

Es el documento en el que se recogen las especificaciones de las pruebas realizadas, donde el trabajador que ejecuta la prueba plasma las anotaciones que considere pertinentes, en función de la posterior corrección por parte de los desarrolladores. La plantilla resumen de pruebas se encuentra en los anexos de este documento.

1.5.2.7 Defecto

Un defecto es un síntoma de un fallo en el software o de un problema descubierto en una revisión, el cual puede ser utilizado para localizar cualquier cosa que los desarrolladores necesiten registrar como síntoma de problema en el sistema.

1.5.2.8 Evaluación de prueba

Es una evaluación de los resultados de los esfuerzos de prueba, tales como la cobertura del caso de prueba, de código y el estado de los defectos. La evaluación es realizada por los diseñadores quienes comparan los resultados obtenidos con los objetivos trazados en el plan de prueba. Durante la evaluación de las pruebas, se realizan métricas que permiten determinar el nivel de calidad del software y qué cantidad de pruebas se deben realizar.

1.6 Equipo de prueba que plantea la metodología RUP.

RUP propone 4 roles fundamentales para los equipos de calidad los cuales son:

1.6.1 Analista de prueba

El Analista de prueba es encargado de identificar y definir las pruebas necesarias para la liberación del proyecto, supervisar el proceso de prueba y los resultados de cada ciclo de prueba. Al finalizar estas etapas evalúa la calidad del proyecto dando los resultados de las pruebas realizadas. Este rol también

representa a los interesados que no tienen una representación directa o regular en el proyecto. Es importante aclarar que solamente evalúa y define las pruebas no las realiza. [3]

Las responsabilidades que tiene asignadas este rol son:

- ✓ Identificar los elementos del destino de la prueba.
- ✓ Definir las pruebas necesarias apropiadas y cualquier dato de prueba asociado.
- ✓ Recopilación y gestión de los datos de prueba.
- ✓ Evaluación del resultado de cada ciclo de prueba.

1.6.2 Diseñador de prueba

El Diseñador de prueba dirige hacia dónde va enfocadas las prueba y garantiza la implementación satisfactoria de las mismas. Esto incluye identificar las técnicas, herramientas y directrices apropiadas para implementar las pruebas necesarias, y para proporcionar orientación al esfuerzo de prueba sobre los requisitos de recursos correspondientes. [3]

Las responsabilidades que tiene asignadas este rol son:

- ✓ Identificar y describir las técnicas de prueba apropiadas
- ✓ Identificar las herramientas de soporte apropiadas
- ✓ Definir y mantener una Arquitectura de automatización de pruebas
- ✓ Especificar y verificar las configuraciones del entorno de prueba necesario
- ✓ Verificar y valorar el enfoque de prueba

1.6.3 Gestor de prueba

El Gestor de prueba dirige el esfuerzo de prueba global, es decir es el que trata de realizar una prueba general del área donde se encuentre. Esto incluye el apoyo de calidad y prueba, la planificación y gestión de recursos y la resolución de cuestiones que impiden el esfuerzo de prueba. [3]

Las responsabilidades que tiene asignadas este rol son:

- ✓ Negociar el objetivo constante y los entregables del esfuerzo de prueba
- ✓ Garantizar la planificación y gestión apropiadas de los recursos de prueba
- ✓ Valorar el progreso y la eficacia del esfuerzo de prueba
- ✓ Apoyar el nivel apropiado de calidad con la resolución de defectos importantes
- ✓ Abogar por un nivel apropiado de comprobabilidad centrado en el proceso de desarrollo de software.

1.6.4 Probador

El Probador o Verificador realiza pruebas y registra los resultados de las pruebas. Es el responsable de documentar todo los defectos encontrados o dar el visto bueno en caso que no encuentre ninguno. [3]

Las responsabilidades que tiene asignadas este rol son:

- ✓ Identificar el enfoque de implementación más apropiado para una prueba determinada
- ✓ Implementar pruebas individuales
- ✓ Configurar y ejecutar las pruebas
- ✓ Registrar los resultados y verificar la ejecución de las pruebas
- ✓ Analizar y recuperar a partir de los errores de ejecución

1.7 Caracterización de Normas y estándares para la realización de pruebas de software.

1.7.1 NC ISO/IEC 9126

La NC ISO/IEC 9126 es una norma cubana que permite especificar y evaluar la calidad del producto de software desde las perspectivas de aquellos asociados con la adquisición, regulación, desarrollo, uso, evaluación, soporte, mantenimiento, aseguramiento de la calidad y auditoria del software. Por ejemplo puede ser utilizada por los programadores, los clientes, el personal de aseguramiento de la calidad y los evaluadores independientes, particularmente los responsables de especificar y evaluar la calidad de los

productos de software. El modelo de calidad definido en esta parte de la NC ISO/IEC 9126 puede usarse para:

• validar la integridad de la definición de los requisitos;
• identificar los requisitos del software;
• identificar los objetivos del diseño del software;
• identificar los objetivos de ensayo del software;
• identificar los criterios de aseguramiento de la calidad;
• Identificar los criterios de aceptación para un producto de software terminado

1.7.2 ISO 9001:2000

La Norma Internacional ISO 9001:2000, Sistemas de gestión de la calidad-Requisitos especifica los requisitos para un sistema de gestión de la calidad. Propone una guía para garantizar los requisitos del cliente y los reglamentarios aplicables, aspirando lograr la satisfacción del cliente a través de la aplicación eficaz del sistema.

La aplicación de estas normas garantiza que todos los requisitos sean genéricos y se pretende que sean aplicables a todas las organizaciones sin importar su tipo, tamaño y producto suministrado. Si dichos productos no pueden aplicar dichas normas son excluidos del mercado.

1.7.3 CMMI

Capability Maturity Model Integration. Modelo para la mejora o evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software. Desarrollado por el Instituto de Ingeniería del Software de la Universidad Carnegie Mellon (SEI), y publicado en su primera versión en enero de 2002.

Los 6 niveles definidos en CMMI para medir la capacidad de los procesos son:

0.- Incompleto: El proceso no se realiza, o no se consiguen sus objetivos.

1.- Ejecutado: El proceso se ejecuta y se logra su objetivo.

2.- Gestionado: Además de ejecutarse, el proceso se planifica, se revisa y se evalúa para comprobar que cumple los requisitos.
3.- Definido: Además de ser un proceso "gestionado" se ajusta a la política de procesos que existe en la organización, alineada con las directivas de la empresa.
4.- Cuantitativamente gestionado: Además de ser un proceso definido se controla utilizando técnicas cuantitativas.
5.- Optimizado: Además de ser un proceso cuantitativamente gestionado, de forma sistemática se revisa y modifica o cambia para adaptarlo a los objetivos del negocio.

1.7.4 SPICE

Marco para métodos de evaluación, no es un método o modelo en sí
Abarca:

- ✓ Evaluación de procesos.
- ✓ Mejora de procesos.
- ✓ Determinación de capacidad.

Alineado con el ISO/IEC 12207.
Intenta proporcionar un marco en el que armonizar los enfoques existentes.

1.7.4 La norma ISO 690 (1 y 2).

- ✓ Está dirigida a autores y editores para la confección de listas de referencia y para la formulación de las citas.
- ✓ Determina los elementos que deben conformar una referencia bibliográfica
- ✓ Determinar el orden de los elementos que componen la referencia
- ✓ Establece las reglas para la transcripción y la presentación de la información de la fuente.

1.8 Herramientas para el entorno de prueba

Existen herramientas que dan soporte a las pruebas de integración**,** pruebas de sistema**,** diagnóstico o afinado de las aplicaciones en los entornos de producción, que generalmente son difíciles de realizar de una forma integrada, comprenden herramientas de análisis estático, automatización de pruebas funcionales**,** de carga**,** de rendimiento**,** de estrés, etc. Estas herramientas son muy utilizadas debido a las ventajas que brindan para el desarrollo de las pruebas de software; algunas de dichas herramientas son:

- WebKing
- QALoad
- SOATest
- JKingQA
- Shadow security scanner
- Jtest
- Apache JMeter.
- JMeter
- Selenium IDE

1.8.1 WebKing de Parasoft

WebKing es una herramienta de pruebas web automatizada, que proporciona pruebas exhaustivas y análisis de los sitios y aplicaciones web para asegurar que cumplen con la fiabilidad, seguridad y metas de desarrollo necesarios para sostener su negocio con eficiencia.

Un modo integrado de sistema múltiple, WebKing de Parasoft señala cuatro áreas de interés:

- ✓ Riesgo del sitio web.
- ✓ Análisis funcional.
- ✓ Pruebas de carga y rendimiento.
- ✓ Análisis de la seguridad.

WebKing mejora la calidad de la aplicación web a través del ciclo de vida del software y mejora la productividad en todo el equipo. Los desarrolladores lo pueden utilizar para efectuar pruebas o análisis web de cada unidad de la aplicación (Servlets, JSP, etc.) y los miembros del equipo de calidad pueden utilizarlo para desarrollar verificaciones de la aplicación. Trabaja sobre las plataformas Windows 2000, Windows Server 2003, Windows XP, Linux y Solaris. **Parasoft**

1.8.2 QALoad de Compuware

QALoad es una herramienta de pruebas para aquellas aplicaciones que estén sometidas a gran carga de usuarios. QALoad ayuda a alcanzar cargas que simulan el comportamiento real del negocio así como a validar que el sistema cumple aceptablemente con los niveles de servicio.

Con QALoad se configura un escenario de pruebas de carga para controlar las condiciones de las pruebas, crear los usuarios virtuales que se necesitan para simular la carga, iniciar y monitorizar las pruebas y se obtienen los informes de resultados. **Parasoft**

1.8.3 SOATest de Parasoft

SOATest proporciona verificación instantáneas de servicios web, simplifica el desarrollo SOA (Service Oriented Architectures), automatiza las pruebas funcionales de cliente/servidor, pruebas de regresión, pruebas de carga/rendimiento y más. SOATest de Parasoft es el estándar de Service Oriented Architectures más aceptado por la industria para su seguridad y fiabilidad. SOATest es una solución probada con clientes como: Yahoo!, Sabre, LexisNexis, IBM, y el gobierno de los Estados Unidos. Las empresas seleccionan SOATest de Parasoft por su gran variedad de funcionalidad, incluyendo validación WSDL four-tier, pruebas unitarias y funcionales del cliente, pruebas unitarias y funcionales del servidor, pruebas de desarrollo así como la habilidad de modelar y probar gráficamente escenarios complejos. SOATest localiza los aspectos fundamentales de los servicios web, como la interoperabilidad, seguridad, cambios de gestión y escalabilidad. SOATest permite que cualquier recurso de prueba creado por un ingeniero pueda ser aprovechado por su equipo de calidad en un escenario de pruebas unitarias y pruebas de carga sin utilizar scripting. SOATest de Parasoft es también la única solución de servicios web

que crea automáticamente pruebas de intrusión de seguridad en servicios web, como inyecciones SQL, inyecciones Xpath, sobrecarga de parámetros, bombas XML y uso de entidades externas. Utilizando SOATest a lo largo del ciclo de desarrollo del software, los usuarios pueden prevenir errores, mejorar la calidad y fiabilidad del producto. **Parasoft**

1.8.4 JKing de ALS

Es una herramienta de análisis estático de código, análisis de dependencias, generación de casos de pruebas y análisis de cobertura pensada para facilitar y automatizar el proceso de adopción de los estándares y pruebas de calidad para un departamento de desarrollo.

ALS ha elaborado con Jking, más de 300 reglas, que ofrecen la capacidad de prevenir los errores y estandarizar el código Java generado de forma automática. La revisión del desarrollo, mediante el análisis de dependencia, y la generación automática de casos de pruebas unitarias harán del código de la aplicación, un código robusto, fiable y bien diseñado. **Parasoft**

1.8.5 JTest de Parasoft

Ofrece varios avances para las industrias desarrolladoras de software de alta calidad. Estos avances tecnológicos están concentrados en la realización de pruebas, para ayudar a los equipos a verificar de manera automática la funcionalidad de aplicaciones cada vez más complejas, en empresas con sistemas en permanente cambio (J2EE, servicios de SOA/Web), todo esto para generar un incremento en la satisfacción del cliente, una reducción en tiempo de entrega del software así como una disminución del riesgo de generar software defectuoso o con problemas de vulnerabilidad. Trabaja sobre las plataformas de Windows 2000, Windows XP, Windows 2003 Server; Solaris y Linux. **Parasoft**

1.8.6 Apache JMeter:

Apache JMeter es un muy complementado banco de pruebas para servidores y aplicaciones web. Las pruebas se configuran jerárquicamente. Los ítems, como parámetros de sesión, acciones y resultados,

aparecen en el árbol del panel izquierdo. Apache JMeter puede efectuar conexiones a bases de datos, servicios como FTP y HTTP o conexiones TCP de otro tipo. Trabaja sobre la plataforma de Win2000/XP/2003/Vista/7. **Parasoft**

1.8.7 Shadow security scanner

Analiza el sistema íntegramente en busca de errores. Es capaz de analizar un rango de IP`s, y luego de hacerlo, brinda un informe en el cual se incluyen los puertos abiertos, agujeros de seguridad, shares compartidos y demás. En cuanto a las vulnerabilidades, produce un informe explicándolas detalladamente, dándoles un nivel determinado de gravedad a cada una, y una posible solución. Incluso posee un algoritmo de seguridad patentado.

Es un completísimo y efectivo escáner de vulnerabilidades para la plataforma de Windows nativa, aunque también examina servidores de cualquier otra plataforma revelando brechas en Unix, Linux, FreeBSD, OpenBSD y Net BSD. También descubre fallos en CISCO, HP y otros equipos de la red. Actualmente, los servicios analizados son: FTP, SSH, Telnet, SMTP, DNS, Finger, HTTP, POP3, IMAP, IRC, Windows Media Service, Terminal Service, NetBIOS, NFS, NNTP, SNMP, Squid, LDAP, HTTPS, SSL, TCP/IP, UDP y servicios del Registro.

Este scanner es modificable por cualquiera que posea algún conocimiento de programación (VC++, C++ o Delphi) pudiendo extender las capacidades del scanner creando auditorías (utilizando el SSS BaseSDK para hacerlo todo más fácil). **Sofonic**

1.8.8 Selenium IDE

Es un plugin de Firefox que pertenece al juego de herramientas SeleniumHQ, permite realizar juegos de pruebas sobre aplicaciones web. Para ello realiza la grabación de la acción seleccionada en un "script", el cual se puede editar y parametrizar para adaptarse a los diferentes casos, y lo que es más importante su ejecución se puede repetir tantas veces como se quiera. El principal objetivo de este plugin es crear pruebas funcionales.

Esta herramienta permite al desarrollador web ahorrarse mucho esfuerzo cada vez que se resuelve alguna incidencia o se genera una versión nueva. Para ello permite automatizar la realización de las pruebas ya sean o bien pruebas específicas o bien juegos de pruebas. **Parasoft**

1.9 Análisis de las estrategias de pruebas creadas en la universidad.

Se realizó un análisis de estrategias realizadas en la universidad con el objetivo de encontrar una que se adecuara a las características del proyecto SAGEB, las cuales se describen a continuación.

1) La estrategia de Pruebas para Juegos de Realidad Virtual que propone la facultad 5 no se adecúa a las necesidades del proyecto SAGEB, ya que no contempla entre los tipos de prueba que realizan las pruebas de Sistema, dígase las Pruebas de Seguridad que son los mecanismos de protección incorporados en el sistema. La prueba de Rendimiento y Resistencia. Estas pruebas son indispensables para las necesidades del proyecto SAGEB por lo que se deben tener en cuenta para la propuesta de estrategia.

Dicha estrategia plantea:

La estrategia de pruebas que se propone consta de listas de chequeo además de la aplicación de 4 niveles de pruebas. Esta estrategia está definida incrementalmente de “adentro” hacia “afuera”, o sea a partir de la prueba de unidad establecida para evaluar el funcionamiento de los módulos, la prueba de integración, la prueba del sistema y por último la prueba de aceptación.

Listas de chequeo:

1. Revisión al Guión Técnico: Aplicar lista de Chequeo propuesta.
2. Revisión a los Requerimientos: Aplicar lista de Chequeo propuesta.

Niveles de Prueba:

1. Pruebas de Unidad: Probar cada módulo por separado.

2. Pruebas de Integración: Probar la interacción de los módulos.
3. Pruebas de Sistema: Verificar que el programa cumple con los requisitos.
4. Pruebas de Aceptación: Validar el producto con el cliente.

En cada nivel las pruebas se realizarán con el método de caja negra utilizando la técnica Partición Equivalente. Para el nivel de unidad se aplicará además el método de Caja Blanca. Las pruebas a aplicar por niveles son:

Pruebas de Unidad
Prueba de Funcionalidad
Pruebas de Caja blanca

Pruebas de Integración
Prueba de Funcionalidad
Pruebas de Sistema
Prueba de Funcionalidad
Prueba de Rendimiento
Pruebas de Interfaz de usuario
Pruebas de Aceptación

2) La estrategia de pruebas de software SCADA Nacional, tampoco es adaptable, ya que no posibilita evaluar exhaustivamente a SAGEB, debido a que esta estrategia no evalúa el software durante todo el ciclo de desarrollo. Mientras para el proyecto SAGEB, se plantea como necesidad la evaluación durante todo el proceso de software.

Esta estrategia plantea:

Se elaboró en conjunto al Plan de Prueba una Lista de Chequeo, relacionados fundamentalmente a la evaluación de la interfaz de usuario del sistema, se trató de abarcar con esto todos los aspectos de usabilidad necesarios, así como los de confiabilidad, eficiencia y portabilidad.

Los tipos de pruebas aplicados en el Sistema SCADA son los siguientes:

• Pruebas Unitarias.

  Pruebas de Funcionalidad.
  Pruebas de Caja Blanca.
• Pruebas Negativas.

  Pruebas de Rendimiento.
  Pruebas de Penetración.

Cada una de estas pruebas se realizó debido a la necesidad de cumplir con los anexos pactados que lo exigían. Algunos de estos anexos incluían pruebas unitarias para algunos módulos en particular, como es el caso del Módulo de Reportes, debido a la importancia que tiene la misma, ya que mediante estas se logra garantizar la ejecución de al menos una vez todos los caminos independientes de cada módulo y se ejerciten todas las estructuras de datos para garantizar su validez.

Entre los aspectos que incluían los anexos pactados se encontraban además las pruebas de penetración al Módulo de Seguridad. Dichas pruebas exigían de una base de la arquitectura del sistema por la cual estaba constituido el SCADA, basándose además en el estudio de la estructura de la red del sistema.

Se decidió aplicar las pruebas de rendimiento al Módulo de Middleware debido a que en un sistema de Supervisión, Control y Adquisición de Datos, el Middleware es la capa de software que se encuentra por encima de los niveles físicos y de transporte y por debajo de las capas de gestión y aplicaciones de usuario. Su función radica en establecer un método de comunicación entre las aplicaciones que se encuentran ejecutándose en cualquier punto del sistema, siendo su localización transparente a las demás.

Se necesitaba verificar el correcto envío y recibo de información por dicho canal de comunicación, y a su vez la eficiencia del tiempo con que debía realizar esta acción según los requerimientos pactados.

Nunca se realizaron las pruebas de integración debido a que para las mismas se necesitaba de un gran número de módulos los cuales terminan probando el sistema en conjunto, lo cual no estaba presente en el SCADA debido a la lejanía existente entre ambos grupos de desarrollo. Este tipo de prueba hubiese sido de gran ayuda ya que la misma contribuye a conocer si los módulos interaccionan entre sí de una forma correcta después de haber sido integrados en el sistema.

Es por ello que tampoco se pudieron llevar a cabo las pruebas de sistema ya que para las mismas se necesitaba que el sistema funcionara como un todo, así como tampoco se efectuaron las pruebas de aceptación por la parte cubana.

3) La estrategia de prueba para el producto Heredia tampoco es adaptable para el proyecto SAGEB, ya que Heredia es un libro virtual el cuál no es el resultado de la integración de módulos de funcionalidad, por lo que los subplanes que tiene serán organizados de acuerdo a la información que se tiene de la aplicación y los tipos de pruebas a realizar. También es válido aclarar que la estrategia del proyecto Heredia no es flexible para adaptarla a otro proyecto ya que propone pruebas diseñadas exclusivamente para el producto Heredia.

Esta estrategia plantea:

Los tipos de pruebas que realizarán son la documentación, validación, usabilidad, instalación y perfil de desempeño.

1.9.1 Conclusiones del análisis:

Como se ha planteado, estas estrategias definidas en la universidad no cumplen con las especificidades que requiere el proyecto SAGEB. Dada la condición que dicho proyecto trabaja en función de la automatización del sistema bancario cubano, deberán redoblarse las fuerzas enfocadas a la seguridad y

funcionalidad del software. Además por lo avanzado que se encuentra el proyecto, la estrategia de pruebas estará orientada solo al método de caja negra.

1.10 Conclusiones Parciales:

En el presente capítulo se realizó un análisis de las diferentes estrategias de pruebas utilizadas y propuestas en la universidad, enunciándose además por qué no se aplican en el proyecto SAGEB. Se definieron una serie de terminologías indispensables para comprender las estrategias de Pruebas.

Además se puede consultar sobre la metodología de desarrollo de software el proceso de prueba, así como de las herramientas a utilizar para la realización de algunas pruebas. Conociendo los conceptos anteriormente definidos, se puede trazar la línea base para definir una estrategia de prueba, donde se lleven a la práctica dichos conceptos, con el fin de reducir o erradicar los errores del software.

Capítulo 2: Propuesta de Estrategia.

2.1 Introducción.

Para obtener un producto de alta calidad es necesario que para la realización de las pruebas se lleve a cabo una buena estrategia de pruebas, dicha estrategia debe abarcar la planificación, la ejecución y los resultados, también se deben tener en cuenta los recursos que se van a emplear tanto humanos como materiales. En este Capítulo se hace la propuesta de una estrategia de pruebas, con el objetivo de maximizar la calidad en el proceso de desarrollo de software del proyecto SAGEB. Tomando como pilar las características específicas del proyecto así como las necesidades priorizables.

2.2 Descripción de la estrategia.

El proyecto SAGEB está compuesto por 37 módulos los cuales están agrupados en 12 subsistemas. Al ser un proyecto complejo, es necesario hacer pruebas que validen la calidad durante todo el proceso de desarrollo de software. Es por eso que el listado de pruebas que se plantea está fragmentado en 4 niveles (Unidad, Integración, Sistema, Aceptación), dentro de los cuales existen una serie de pruebas, que abarcan la validación de todo el software, todas las pruebas están orientadas al método de caja negra. Además están planteadas de lo sencillo a lo complejo (incrementalmente).

En la estrategia propuesta se deben realizar a cada tipo de prueba tres iteraciones aunque a medida que pasan las iteraciones las probabilidades de detectar no conformidades disminuyen no se pueden descartar, pues se debe asegurar un producto con óptimos resultados, que validen íntegramente el éxito de la prueba. Luego de realizar la primera iteración se deben obtener las no conformidades, que serán entregadas a los desarrolladores, para su posterior corrección en un plazo de 10 a 15 días según la complejidad de las no conformidades. Es válido señalar que en el proyecto SAGEB los analistas son los responsables de revisar y corregir los diseños de caso de pruebas elaborados y además en coordinación con los usuarios finales emiten juegos de datos que es el utilizado por los probadores para poner a prueba

el sistema. Una vez concluida las pruebas se registran las pruebas en el redmine, asignadas al analista principal y este se encarga de distribuirlas a su equipo.

2.2.1 Para cada iteración se debe:

- ✓ Configurar el entorno de prueba.
- ✓ Identificar los objetivos y los artefactos que se entregan de las pruebas.
- ✓ Aplicar estrategia de prueba en el nivel que se encuentre.
- ✓ Resultado de la prueba (No conformidades encontradas y resúmenes de pruebas).

Esta estrategia de pruebas define los procedimientos en cuanto a los siguientes aspectos:

- ✓ ¿Qué metodología de pruebas se va a usar?
- ✓ ¿Cuál será el entorno de pruebas?
- ✓ ¿Cómo debe estar conformado el equipo de calidad?
- ✓ ¿Qué herramientas de pruebas ser a usarán?
- ✓ ¿Qué técnicas específicas de pruebas serán usadas para probar el software?
- ✓ ¿Cómo serán tratados los riesgos en el proceso de prueba?

2.3 Metodología de pruebas.

El problema del software se reduce a la dificultad que afrontan los desarrolladores para coordinar las variadas cadenas de trabajo de un proyecto. La comunidad desarrolladora de software necesita una forma conjugada de trabajar. Necesita un método común de trabajo, que integre las múltiples facetas del proceso de desarrollo del software, un proceso que:

- ➢ Proporcione una guía para ordenar las actividades de un equipo.
- ➢ Especifique los artefactos que deben desarrollarse.
- ➢ Dirija las tareas de cada desarrollador por separado y en equipo.
- ➢ Ofrezca criterios para el control y la medición de los productos y actividades del proyecto.

Para la elaboración de esta estrategia se utilizará la metodología RUP, debido a que ésta es la metodología que se usa en el proceso de desarrollo del proyecto SAGEB.

2.4 Entorno de pruebas.

El entorno de prueba se compone por todos los activos fijos tangibles, que tenga a su disposición el equipo de calidad del proyecto. Estos recursos varían en dependencia de las necesidades reales que presenten el proyecto.

También forman parte del entorno de prueba los recursos humanos del proyecto, ya que los trabajadores son la pieza fundamental en el proceso de desarrollo de software. Gestionar los recursos humanos del proyecto, es un paso necesario para la estrategia que se plantea.

Para el proyecto SAGEB se definen 6 máquinas clientes –atendiendo a la cantidad de recursos humanos existentes en el proyecto y la cantidad de equipos de trabajo que fueron creados- estas deben estar conectadas mediante un Switch al servidor de aplicaciones a través del protocolo http y este a su vez se conecta con el servidor de base de datos, para garantizar el acceso a la aplicación sin problemas. Cada uno de los servidores tiene un servidor de respaldo, en caso de presentarse algún problema.

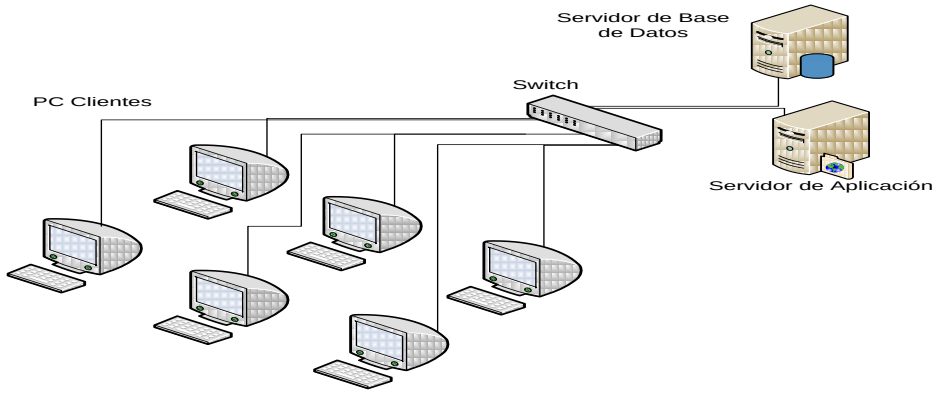

Figura 2: Entorno de Trabajo.

2.5 Grupo de calidad SAGEB

A partir de los roles que plantea RUP el grupo de calidad de SAGEB será conformado por un total de 12 personas, atendiendo a que el proyecto tiene 183 casos de uso, que los diseñadores tienen categoría media y demoran aproximadamente 45 minutos en realizar un diseño de caso de prueba y generalmente un probador actualiza un diseño de caso de prueba y registra las no conformidades en el redmine en 1 hora. Los cuales ocuparían los siguientes roles:

- **Administrador de calidad** quien responde por el trabajo del equipo y organiza el cumplimiento del plan de pruebas y la correcta implementación de la estrategia de prueba. Es una persona bien orientada, su objetivo principal se centra en asegurarse que la aplicación se ajusta a los requerimientos y mantenerse lo más lejano posible de errores. Evalúa los resultados que se obtienen al realizar las pruebas de calidad.
- **Tres diseñadores de prueba** con total dominio sobre las técnicas de prueba. Diseña los casos de pruebas. Valora y documenta el efecto de las pruebas realizadas al producto. Define las listas de chequeo.
- **Dos analistas de pruebas** quienes deben tener total dominio de las características del sistema creado en el proyecto SAGEB. Identifican y definen las pruebas requeridas. Monitorean el progreso de las mismas y el resultado en cada ciclo de prueba. Evalúan la calidad total experimentada como un resultado de las actividades de prueba.
- **Seis probadores** quienes tiene gran responsabilidad, ya que de ellos depende un trabajo con el nivel de calidad requerido. Realizan las pruebas especificadas en el plan y el registro del resultado de las mismas.

Los integrantes del grupo de calidad no pueden ser parte de otros grupos del mismo proyecto, es decir no pueden ser programadores, ni diseñadores de SAGEB, con el objetivo de que las pruebas se realicen netamente de forma independiente, el trabajo de los probadores no puede ser influenciado por otros roles que propicien otros puntos de vista.

2.6 Herramientas de prueba

2.6.1 Herramientas manuales

Casos de pruebas

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada.

Los casos de pruebas deben verificar:

- ✓ Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificaciones de los requerimientos.
- ✓ Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

Después de realizar la primera iteración de pruebas se obtienen las no conformidades las cuales se pueden clasificar en:

- ✓ Significativa
- ✓ No significativa
- ✓ Recomendación

Luego de darle tratamiento a estas no conformidades obtenidas, para la segunda iteración ya estarían clasificadas de acuerdo con los resultados del tratamiento dado, de esta forma se clasificarían en:

- ✓ Resueltas
- ✓ No procede
- ✓ Pendiente de solución

2.6.2 Herramientas automatizadas

Para comenzar el tema de las pruebas automatizadas es necesario conocer una serie de ventajas y desventajas que acarrea el uso de este recurso, con el objetivo de seleccionar o no diferentes alternativas y vías de solución a problemas que puedan existir. A continuación se verán algunas de ellas:

Ventajas:

- ✓ Rapidez en la ejecución de pruebas.

- ✓ La ejecución de un mayor número de pruebas en una unidad de tiempo finita. Esta característica hace que el producto final sea mucho más confiable y por supuesto tenga mucha más calidad.

- ✓ Simulación de condiciones reales de explotación. Existen pruebas de vital importancia que no es factible realizarlas de forma manual.
- ✓ Programación de la ejecución de pruebas en horarios no laborales como la noche y los fines de semana. Muchas herramientas brindan prestaciones que permiten la programación automática de un sinnúmero de pruebas, utilizando así el horario de trabajo para idear posteriores planes de prueba.

Desventajas:

- ✓ Por lo general una gran cantidad de aplicaciones contienen elementos que no son compatibles con las herramientas que se utilizan para ponerlas a prueba, en consecuencia esto requiere la búsqueda de soluciones creativas para que éstas se adapten a la aplicación, lo cual constituye un obstáculo al que se tendrá que enfrentar el equipo de trabajo.

- ✓ Poco conocimiento de lenguajes de scripting por parte de los probadores. Éste puede ser un punto determinante en el proceso de automatización de pruebas pues una gran cantidad de herramientas utilizan estos lenguajes para el mantenimiento de las mismas.

- ✓ Si se automatiza el caos sólo se logra un caos mucho peor. No es recomendable la práctica de las pruebas automáticas por equipos que tengan mal organizada la realización de éstas. Ejemplos de malas prácticas son: poca o inconsistente documentación, pruebas que no son muy efectivas en la búsqueda de defectos, etcétera.

- ✓ Problema del mantenimiento. Cuando los sistemas sufren algún tipo de cambio, como ya se ha visto anteriormente, las pruebas también tienen que cambiar. El esfuerzo empleado en la actualización de las mismas es a veces causa de abandono de la iniciativa de su automatización y de un retorno a la ejecución manual.

A partir de la investigación realizada se definió que solo se implementarán con herramientas automatizadas las pruebas que resulte muy engorroso realizarlas manualmente. Del listado de herramientas analizadas se ha hecho especial énfasis en las libres y de código abierto por el ahorro económico que representan y las posibilidades de personalización a los intereses tecnológicos del centro. Las herramientas seleccionadas son: JMeter y Shadow Security Scanner, ya que de estas herramientas existe además una amplia documentación sobre su modo de empleo y los requerimientos no funcionales que ella precisa.

JMeter

Se caracteriza por sus funcionalidades para realizarle pruebas de estrés, carga y rendimiento a las aplicaciones de manera independiente, al permitir aislar los subsistemas de la aplicación. Además puede ser configurable porque permite definir la cantidad de usuarios que va a simular y las pruebas que se le aplican al marco de trabajo.

De forma general para lograr un mejor entendimiento de su funcionamiento se desglosan a continuación algunas características. (Ver anexos 6)

- ➢ Se comporta como un proxy local para que a través de él transite toda la información en tiempo real.

- Permite analizar su rendimiento, dadas determinadas condiciones como la navegabilidad en el sistema.

- Numera los errores ocurridos durante la transferencia de datos del servidor al navegador, muchos de estos errores no son visibles para el cliente pero pueden provocar dificultades a corto o largo plazo.

- El muestreo de la información a través de tablas y gráficas facilita la comprensión del flujo de datos y sus características.

- Además brinda información sobre el tiempo de servicio al sistema, el por ciento de rendimiento, el tráfico de información entre el cliente y el servidor tanto los correctos como los incorrectos.[4]

Shadow Security Scanner

La herramienta Shadow Security Scanner permite escanear la red en busca de vulnerabilidades lo cual ayuda a evaluar el nivel de seguridad que posee el sistema a través de la red y a su vez evita el surgimiento de algún inconveniente que ponga en riesgo el adecuado funcionamiento del sistema. (Ver anexo 5)

2.7 Actividades definidas para el proceso desarrollo de cada prueba.

Las siguientes actividades que se definen son pasos que se proponen para las prueba que integran la estrategia. Las actividades planificar, diseñar y ejecutar prueba son comunes en cada nivel, sin embargo la actividad implementar prueba solo se realiza en el nivel de Sistema debido a que en este nivel se realizan pruebas automatizadas que generan componentes.

2.7.1 Actividad: Planificar prueba.

El propósito de planificación de la prueba es planificar los esfuerzos de prueba en una iteración llevando a cabo las siguientes tareas:

- ✓ Describir una estrategia de prueba.

- ✓ Estimando los requisitos para el esfuerzo de una prueba (recursos humanos y sistemas necesarios).
- ✓ Planificar el esfuerzo de una prueba.

Los desarrolladores de pruebas desarrollan una estrategia de prueba para iteración donde especifican que tipo de pruebas ejecutar, cómo y cuándo ejecutar dichas pruebas.

2.7.2 Actividad: Diseñar prueba

Los propósitos de diseñar las pruebas:

- ✓ Identificar y definir los casos de pruebas para cada construcción.
- ✓ Identificar y estructurar los procedimientos de pruebas especificando cómo realizar los casos de pruebas.

2.7.3 Actividad: Implementar Prueba

Es automatizar los procedimientos de prueba creando componentes de prueba. Esta es una actividad que no se realiza en todas las pruebas, solo en las pruebas que son automatizadas.

2.7.3 Actividad: Ejecutar prueba

La forma de ejecución de cada una de las pruebas es especificada más adelante en este capítulo.

2.8 Propuesta de pruebas.

A continuación se describe cada una de las pruebas que define la estrategia para cada nivel.

2.8.1 Pruebas de Unidad

Se realizarán pruebas de unidad para cada uno de los 37 módulos del proyecto SABEB, de manera independiente. Dentro de este nivel se realizarán las pruebas de funcionalidad y de usabilidad.

Estas pruebas se realizarán durante la fase de elaboración a la par del proceso de implementación se irán diseñando los 183 casos de prueba, en un estimado de 17 a 18 días al tener en cuenta que los diseñadores tienen un promedio aproximado de 480 min laborables del día entre los 45 min de diseño de caso de prueba da un resultado de 10.6 casos de uso por día laborable; si los 183 casos de prueba entre los 10.6 casos de pruebas da un resultado de 17.2 días aproximadamente. Los probadores son los responsables de ejecutar las pruebas, actualizar los diseños de caso de prueba y emitir las no conformidades encontradas en un estimado de 22 a 23 días al tener en cuenta que los probadores trabajan 1 hora por cada funcionalidad a probar al tener 8 horas de trabajo en el proyecto serán 183 funcionalidades a probar entre las 8 por día serán 22.3 días aproximadamente. Los días de trabajo no son consecutivos debido a las posibles afectaciones que pudieran presentarse por enfermedad, exámenes, reuniones, días no laborables, entre otras. Dentro de este nivel se proponen pruebas de funcionalidad y de usabilidad las cuales se ejecutarán simultáneamente, es decir se deben verificar las funcionalidades y a la vez verificar no conformidades referentes a estética.

Las pruebas de unidad están previstas en 3 iteraciones para cada módulo en cada tipo de prueba.

1ra Iteración: Se realizan los diseños de Caso de Pruebas por descripciones de casos de uso. Se prueban los módulos. Se emiten los resultados con las no conformidades. Se distribuyen esas no conformidades, y tienen un plazo mayor de 10 días y menor de 15 para realizar el tratamiento de las no conformidades.

2da Iteración: Se realizan las revisiones de las no conformidades detectadas en la 1 iteración. Se emiten los resultados con las no conformidades; se actualiza en el redmine el caso de prueba. Se distribuyen las no conformidades a los desarrolladores, y tienen un plazo mayor de 10 días y menor de 15 para realizar el tratamiento de las no conformidades.

3ra Iteración: Con menos probabilidades de encontrar errores pero tratando de lograr una mayor confiablidad se vuelven a realizar todo el proceso de pruebas y búsqueda de no conformidades.

A medida que van saliendo los módulos de la fase de implementación se les realizarán las:

2.8.1.1 Pruebas de Funcionalidad:

Descripción: Estas pruebas se aplican para verificar que el módulo se adecúa a los requerimientos funcionales. Es importante decir que se realizarán a cada uno de los módulos del proyecto, a medida que estos vayan saliendo.

Objetivo: Ejecutar los métodos de los módulos bajo distintas condiciones o distintas variables de entrada y analizar el resultado.

Responsable de ejecutar la prueba: Probador

Responsable de diseñar la prueba: Diseñador de Pruebas.

Comienzo de la prueba: Basada en lo que propone la metodología RUP debe comenzar durante la Fase de elaboración del producto, a los dos días de estar terminado el primer módulo. Para poder dar comienzo a dichas pruebas es necesario que estén capacitados los probadores para que sean realizadas correctamente y laboratorio de calidad debe estar apto para poder comenzar el trabajo.

Procedimiento para realizar la prueba: Se realizarán pruebas de funcionalidad (explicadas en el capítulo anterior) a cada uno de los 37 módulos de forma independiente, las cuales iniciarán con los juegos de datos proporcionados por el cliente y los casos de pruebas basados en los casos de uso creados por los analistas del proyecto. El diseñador confecciona los casos de prueba y el líder de calidad los revisa. La información generada durante el proceso de prueba será guardada en el expediente de proyecto en Implementación y Prueba. Una vez diseñados y aprobados los casos de prueba el probador los ejecuta y documenta los resultados obtenidos, si se detecta alguna no conformidad, estas deben ser registradas en la tabla de no conformidades que se encuentra en la plantilla de Diseño de Casos de Prueba y deben ser especificadas en el resumen de prueba (Ver plantillas en anexos).

2.8.1.2 Pruebas de Usabilidad

Descripción: Estas prueba se enfocan en trabajar con los factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento.

Objetivo: Asegurar el éxito y la satisfacción del usuario.

Responsable de ejecutar la prueba: Probador

Responsable de diseñar la prueba: Diseñador de Pruebas.

Comienzo de la prueba: Basada en lo que propone la metodología RUP debe comenzar durante la Fase de elaboración del producto, simultáneamente con las pruebas de funcionalidad. Para poder dar comienzo a dichas pruebas es necesario que estén capacitados los probadores para que sean realizadas correctamente y laboratorio de calidad debe estar apto para comenzar el trabajo.

Procedimiento para realizar la prueba: Se realizarán pruebas usabilidad a cada uno de los 37 módulos de forma independiente, las cuales iniciarán con los casos de pruebas diseñados los cuales estarán enfocados principalmente a las características de interfaz de usuarios. El diseñador confecciona los casos de prueba y el líder de calidad los revisa. La información generada durante el proceso de prueba será guardada en el expediente de proyecto en Implementación y Prueba. Una vez diseñados y aprobados los casos de prueba, el probador los ejecuta y documenta los resultados obtenidos, si se detecta alguna no conformidad, estas deben ser registradas en la tabla de no conformidades que se encuentra en la plantilla de Diseño de Casos de Prueba y deben ser especificadas en el resumen de prueba (Ver plantillas en los anexos).

2.8.1.3 Pruebas de integración.

Descripción: Las pruebas que se realizan en este nivel son de tipo de desarrollador, ya que los propios programadores son los encargados de comprobar que el módulo en el que están trabajando no tenga problemas para integrarlo al subsistema. Estas pruebas se realizarán a la par de las correspondientes al nivel de Unidad.

Objetivo: Son las pruebas para probar cómo los componentes construidos funcionan en conjunto.
Responsable de ejecutar la prueba: Desarrollador del Sistema y el Arquitecto del proyecto.
Comienzo de la prueba: Estas pruebas comienzan durante la Fase de elaboración del producto, a medida que los desarrolladores vayan terminado los módulos que conforman los subsistemas.
Procedimiento para realizar la prueba: En el proyecto SAGEB se utiliza el Subversion que no es más que un software de sistema de control de versiones. El Subversion da la posibilidad de poder acceder al repositorio a través de redes, permitiendo que los desarrolladores puedan ir modificando indistintamente a la misma vez desde cualquier punto de trabajo en los laboratorios del proyecto. Trayendo como ventaja poder progresar rápidamente en la implementación de software y tener el control de los cambios que realicen los desarrolladores. Los desarrolladores una vez que suben los cambios que realizan a los módulos el sistema reconoce cuando hay error a la hora de integrar los módulos. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada.
Por otra parte se realizarán pruebas independientes, las cuales las realizará el arquitecto del proyecto SAGEB, ya que tiene los conocimientos necesarios para validar que la integración de los módulos no tenga ningún defecto para llegar al producto final con los resultados esperados.

2.8.2 Pruebas de validación parcial.

Estas pruebas son internas, se realizan por los funcionales de la entidad cliente, en este caso Banco Nacional de Cuba (BNC), en la fase de Elaboración. Estas pruebas estarán divididas en dos bloques (bloque 1: primer 50% terminado (es decir que ya hayan terminado las pruebas de unidad) y bloque dos: el resto de los módulos), la primera se planifica en el momento que se terminen las pruebas de unidad al primer 50% de los módulos terminados y una segunda parte que se realizaría con el resto de los módulos. Los clientes proporcionarán los Juegos de Datos para probar el sistema, estas pruebas dispondrán de 2 Iteraciones por cada bloque. Es importante destacar que para realizar estas pruebas no es necesario que se hayan culminado con el nivel de unidad, solo con el 50 % terminado se procede a esta prueba.

Luego de la 1ra iteración tendrá un plazo de 7 días para que los desarrolladores corrijan las no conformidades encontradas por el equipo de prueba en conjunto con los funcionales.

Ya en la segunda iteración se revisan las NC detectadas en la 1 iteración, en esta iteración los defectos encontrados, pero en caso de que se encontrarán los desarrolladores tendrán 7 días para solucionarlos.

Estas pruebas de validación parcial garantizan que el cliente vaya monitoreando el estado del software, pero además son un medidor importante antes de realizar las pruebas de Aceptación y el despliegue del proyecto porque permite ver y corregir errores antes que el producto este terminado completamente.

2.8.3 Pruebas de Sistema

Este nivel está destinado a verificar el programa final. Al ser SAGEB un producto estrictamente dirigido al trabajo bancario, es necesario comprobar su capacidad de respuesta ante hecho extremos. Las pruebas que se proponen en este nivel son todas con herramientas automatizadas. Debido a que en el nivel de unidad se comprobaron las funcionalidades de forma independiente y en el nivel de integración se comprobó que integrados funcionen correctamente, en este nivel se enfocan las pruebas a los requerimientos no funcionales que son de vital importancia para el producto.

Para las pruebas de sistema se planifican 3 iteraciones:

1ra Iteración: Se prueba el sistema, que debe estar funcionando como un todo. Se emiten los resultados con las no conformidades. Se distribuyen esas no conformidades, y tienen un plazo mayor de 10 días y menor de 15 para realizar el tratamiento de las no conformidades.

^{2da} Iteración: Se realizan las revisiones de las no conformidades detectadas en la 1 iteración. Se emiten los resultados con las no conformidades. Se distribuyen esas no conformidades, y tienen un plazo mayor de 10 días y menor de 15 para realizar el tratamiento de las no conformidades.

3ra Iteración: Con menos probabilidades de encontrar errores pero tratando de lograr una mayor confiablidad se vuelven a realizar todo el proceso de pruebas y búsqueda de no conformidades.

Estas pruebas se realizarán en la fase de Construcción para garantizar aspectos fundamentales en un tipo de software como el que se implementa en el proyecto SAGEB, como la seguridad, la resistencia y el rendimiento, las cuales se detallan a continuación:

2.8.3.1 Prueba de Seguridad.

Descripción: Se verifican los mecanismos de protección incorporados en el sistema.

***Objetivo*:** Verificar que los mecanismos impidan los accesos impropios.

Responsable de ejecutar la prueba: Probador.

Responsable de diseñar la prueba: Diseñador de Pruebas.

Comienzo de la prueba: Basada en lo que propone la metodología RUP debe comenzar durante la Fase de construcción del producto.

Procedimiento para realizar la prueba: Las pruebas de seguridad están enfocadas hacia la seguridad de la red del sistema. Para la ejecución de dichas pruebas se propone el uso de la herramienta automatizada Shadow Security Scanner que en la siguiente tabla se describe el procedimiento para su implementación:

Tabla 2 Procedimiento Shadow Security Scanner para seguridad.

Descripción del escenario para pruebas de seguridad con Shadow Security Scanner	
Precondiciones	• Se crea una nueva sesión para escanear. • Se escoge el tipo de escáner a ejecutar. • Se llenan los parámetros requeridos para el éxito del escaneo.

	• Se procede a escanear la red.
Entrada	• Se agregan los host/ip a analizar.
Procedimientos	• La herramienta indica el host, junto con los puertos, auditorías, servicios, recursos compartidos y usuarios.
Pos condiciones	• Describe las características de los puertos abiertos encontrados.
Salida	• Se brinda la información relacionada al host especificado. • Se especifica la gravedad de la vulnerabilidad encontrada según los colores : – Verde: Bajo Riesgo. – Celeste: Medio Riesgo. – Rojo: Alto Riesgo.
Análisis de resultados	• Permite determinar la seguridad de la red donde se encuentra instalado el sistema. • Se muestran las irregularidades que puedan existir en la red especificada así como los puertos abiertos que presenta.

2.8.3.2 Prueba de Resistencia.

Descripción: Son dedicadas a someter al sistema a una carga tan grande de usuarios que sea capaz de colapsar el mismo.

***Objetivo*:** Evaluar la capacidad de resistencia ante un elevado número de usuarios.

Responsable de ejecutar la prueba: Probador.

Responsable de diseñar la prueba: Diseñador de Pruebas.

Comienzo de la prueba*:* Basada en lo que propone la metodología RUP debe comenzar durante la Fase de construcción del producto.

Procedimiento para realizar la prueba: Existe una gran similitud entre las pruebas de resistencia y las pruebas de rendimiento, se diferencian en los objetivos de cada una. La herramienta automatizada Jmeter se utiliza para la implementación de ambas pruebas, solo cambia el enfoque pues para realizar la prueba de rendimiento se propone incrementar el número de usuarios a un nivel elevado, para lograr sobrecargar la aplicación y verificar su nivel de resistencia. La siguiente tabla explica el procedimiento para la implementación de este tipo de prueba.

Tabla 3 Procedimiento de JMeter para resistencia

Descripción del escenario para pruebas de resistencia con JMeter	
Precondiciones	• Se instala la maquina virtual de java. • Se busca la aplicación a la cual se le van a realizar las pruebas. • Se configura el JMeter como servidor proxy. • Se configura el navegador como: localhost y se arranca el servidor proxy. • Se inicia la navegación por el sitio deseado.

	• Se registra todo el tráfico desde el servidor hacia el cliente. • Se guarda los resultados de la navegación.
Entrada	• Se seleccionan la cantidad de usuarios a simular. Esta entrada debe exceder el número promedio de usuarios que pueden conectarse. • Se específica el tiempo que va a demorar la conexión
Procedimientos	• Se especifican los listeners para registrar los datos acorde a los objetivos. • Se arranca la prueba.
Pos condiciones	• Los listeners registran el mismo tráfico previamente realizado con la cantidad de usuarios definidos.
Salida	• En los listener las tablas y las graficas se muestran el tiempo de respuesta del sistema bajo las condiciones configuradas. • Se muestran los componentes enviados desde el servidor así como los errores que se produjeron. • Informa del porciento de rendimiento.
Análisis de resultados	• Permite determinar si los tiempos de respuesta son los esperados bajo determinadas condiciones. • Muestra si la aplicación está trabajando con todas sus funcionalidades de forma correcta. • Los resultados en porcientos muestra si los componentes operan de forma correcta.

2.8.3.3 Prueba de Rendimiento

Descripción: Está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

Responsable de ejecutar la prueba: Probador.

Responsable de diseñar la prueba: Diseñador de Pruebas.

Comienzo de la prueba: Basada en lo que propone la metodología RUP debe comenzar durante la Fase de construcción del producto.

Procedimiento para realizar la prueba: Para la realización de las pruebas de rendimiento se propone la utilización de la herramienta automatizada JMeter, de la cual se describen los procedimientos para su utilización en las pruebas de resistencia.

2.8.4 Pruebas de Aceptación.

Descripción: Es la prueba final antes del despliegue del sistema.

***Objetivo*:** Verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido

Responsable de ejecutar la prueba: Cliente y la comprueba el Analista.

Responsable de diseñar la prueba: Diseñador de Pruebas.

Comienzo de la prueba: Antes de realizar el despliegue del software.

***Procedimiento para realizar la prueba*:** Los errores y cambios que se soliciten quedan recogidos en las plantilla de no conformidades del expediente de proyecto. En caso de no encontrar errores el cliente deja corroborado la aceptación del producto. Y se debe de validar la posibilidad de cambios. (Ver plantillas en anexos del documento)

2.9 Tratamientos de riesgos.

Detectados algunos de los riesgos que pueden surgir en el proceso de implementación de la estrategia de prueba planteada, se propone:

Tabla 4: Riesgos.

Riesgos	Tratamiento
Que no sean corregidos los defectos en el tiempo planificado, es decir que los desarrolladores excedan el tiempo previsto para la rectificación, lo que puede traer como consecuencia una alteración en la planificación de las pruebas.	En caso de que los errores no sean corregidos los defectos en el tiempo planificado, se responsabiliza al líder de proyecto de que se cumpla en el tiempo mínimo.
Que en el nivel de aceptación existan no conformidades por parte del cliente, lo que puede conllevar cambios en los requerimientos del software.	En caso que en el nivel de aceptación existan no conformidades por parte del cliente, se analiza si es error en la implementación de los desarrolladores o si esta en los requisitos del sistema, en el primer caso los desarrolladores deben solucionar el problema en el menor tiempo estimado, en el segundo caso se debe empezar la rectificación desde los requerimientos.

Que no sean aplicadas correctamente las pruebas por parte de los probadores.	Para prever el riesgo que no sean aplicadas correctamente las pruebas por parte de los probadores, el líder de calidad debe hacer un chequeo periódico, tomando una muestra del trabajo independiente de cada probador y validar que se esté trabajando correctamente.
Que no sean especificadas correctamente las no conformidades o el resumen de prueba, podría contribuir a confusiones en los desarrolladores.	Que no sean especificadas correctamente las no conformidades o el resumen de prueba, se puede controlar dándole al diseñador la tarea de revisar muestras de trabajos de cada probador.
Que el personal de calidad no esté capacitado para cumplir correctamente sus funciones.	Para prever que el personal de calidad no esté capacitado, es necesario que se les imparta un curso de capacitación sobre el rol que debe desempeñar en el proyecto.
Que no existan los recursos necesarios para cumplir con los requisitos del entorno de prueba.	En caso que no existan los recursos necesarios para cumplir con los requisitos del entorno de prueba, el líder de calidad debe informar al líder de proyecto para que éste busque las condiciones precisas, ya que las pruebas no pueden realizarse sin el entorno adecuado.

Inexperiencia del equipo de aseguramiento de calidad.	Impartir cursos de capacitación para los aseguradores de calidad. Auto preparación por parte del equipo de calidad.
Poca atención por parte del cliente en las reuniones o encuentros para el análisis del proyecto.	El cliente debe firmar un contrato donde esté plasmado que él debe dedicarle tiempo a las reuniones de proyecto. Informarle al cliente que debe dedicarles tiempo al proyecto y participar en las reuniones de proyecto.

2.10 Conclusiones parciales

La estrategia de prueba se puede visualizar como el plan estratégico para efectuar pruebas contra uno o más aspectos de los sistemas. Teniendo como objetivos fundamentales, comunicar la estrategia a los interesados para obtener el acuerdo sobre el enfoque que se le debe dar a la misma; comunicar la estrategia a los miembros internos del equipo de prueba para habilitar un esfuerzo de equipo coordinado; convencer a los gestores y a otros interesados de que el enfoque es acertado y alcanzable. Conociendo estos objetivos de la estrategia de prueba, se considera que la propuesta realizada en este capítulo es la adecuada para obtener un sistema bancario de calidad.

Capítulo 3" Resultados Obtenidos"

3.1 Introducción

Atendiendo a que la estrategia de pruebas guía el equipo de un proyecto de desarrollo de software en cuanto a la realización de las pruebas, esta se encuentra ampliado en el capítulo 1 con la evaluación de otras estrategias y materializado en el capítulo 2.

En este capítulo 3 se realiza la comprobación y aplicación de la estrategia de pruebas propuesta a partir de los resultados obtenidos en su aplicación en el proyecto SAGEB, así como los avales de la Asesora de Calidad del Centro y la líder del proyecto. Atendiendo a que es la única forma de comprobar los resultados obtenidos hasta la fecha con la utilización de la misma.

3.2 Resultados obtenidos

El equipo de calidad del proyecto SAGEB, se encuentra aplicando la estrategia de pruebas planteada en el Capítulo 2. A partir de los procedimientos expuestos en esta estrategia ya se ha implementado el 50% de pruebas correspondientes al nivel de unidad y las dos iteraciones al primer bloque de las pruebas de validación parcial.

El sistema cuenta con 216 requerimientos funcionales de los cuales se han probado los 125 correspondientes a los 19 módulos que representan un 50% de los módulos del proyecto. A partir de estos requerimientos se realizaron los diseños de casos de pruebas para la implementación de las pruebas de funcionalidad en el nivel de unidad, con el objetivo de que el sistema dé cumplimiento a cada una de las funcionalidades especificadas por el cliente, simultáneamente a las pruebas de funcionalidad se ejecutaron las pruebas de usabilidad.

El grupo de calidad del proyecto SAGEB ha completado las 3 iteraciones de las pruebas de funcionalidad y usabilidad y las 2 iteraciones de las pruebas de Validación parcial con el cliente.

La aplicación de las pruebas arrojaron los siguientes resultados

Tabla 5: Resultados obtenidos.

Subsistema	Módulo	1ra iteración Unidad		2da Iteración Unidad		3ra iteración Unidad		Validación parcial	
			Total NC del Subs		Total NC del Subs		Total NC del Subs	1ra	2da
Carta de crédito	Gestionar acuerdo	4	16	1	6	-	1	5	1
	Documentos de embarque	5		3		1		6(1 SC)	2(SC)
	Permiso sobre cuenta	4		1		-		-	1
	Capacidad financiera	3		1				-	
Contabilidad	Plan de cuentas	5	33	2	15	1	5	5(5 SC)	1(1 SC)
	Gestionar Banco	2		1		-			
	Gestionar Comisiones	8		4		2		1	1
	Libros Contables	1		-		-		2	
	Transacciones generales	5		2		-		3	1
	Tasa de Cambio	2		1		-		1	

	Cuentas de Banco	1		-		-		1	1
	Gestionar cuentas concepto	3		2		1		-	
	Cierre contable	5		1				-	
	Inicio contable	2		2		1		-	
Clientes	Clientes Jurídicos	1	2	1	3	-	-	2	
	Personas Autorizadas	1		2		-		5	2
Módulos comunes	Medios de comunicación	8	8	4	4	2	2	1	1
Cuentas de clientes	Gestionar cuenta	5	8	1	3		1	2	1
	Reservación de fondo	3		2		1		-	
Total NC			67		31		9	34(6 SC)	13(3 SC)

Como se muestra en la tabla anterior durante la primera iteración de las pruebas de unidad se obtuvieron 67 no conformidades, antes de la segunda iteración ya se había solucionado el 70.1% (47 NC) como se muestra en la siguiente figura.

Figura 3: Resultados 1ra Iteración.

Además se puede observar que en la segunda iteración se obtuvieron un total de 31 no conformidades de las cuales se solucionaron antes de la tercera iteración al 67.7% (21 NC), así se encuentra reflejado en la siguiente figura.

Figura 4: Resultados de la 2da Iteración.

En la tercera iteración se obtuvieron 9 no conformidades de las cuales 8 fueron resueltas y 1 no procede.

Figura 5: Resultados de la 3ra Iteración.

Se realizaron también las pruebas de validación parcial las cuales ya arrojaron sus resultados, las mismas fueron realizadas siguiendo lo estipulado en el plan de prueba. Durante la primera iteración de estas pruebas se encontraron 27 no conformidades y 6 solicitudes de cambios. Durante la segunda iteración los clientes encontraron 13 no conformidades y 3 solicitudes de cambio, de ellas 2 críticas para el sistema. Actualmente se corrigieron las no conformidades encontradas y el proyecto ya entregó los módulos probados al centro de calidad para soluciones tecnológicas (Calisoft) existente en la universidad.

3.3 Conclusiones parciales

Los resultados alcanzados muestran de forma parcial que la estrategia planteada es adecuada para el proyecto SAGEB, como consecuencias de las características específicas del proyecto y la estrategia fue concebida especialmente para dicho proyecto. Hasta el momento se obtuvieron resultados parciales, donde como se muestra en las siguientes figuras las pruebas de unidad redujeron el numero de no conformidades al 33.3% esto demuestra que la manera en que están orientadas las pruebas, fortalece el trabajo de calidad en el proyecto brindando mayor seguridad.

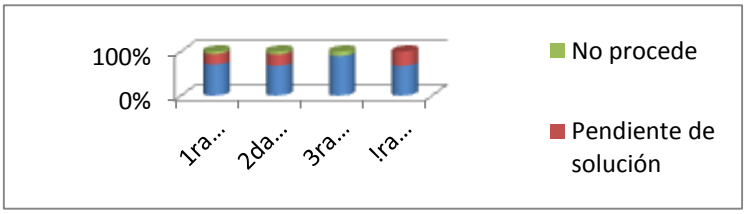

Figura 6: Resultados generales.

Figura 7: Resultados.

La elaboración de un producto de calidad es un gran logro que requiere un esfuerzo continuo, alcanzar la calidad absoluta es un gran objetivo que siempre se debe trazar. Como resultado del presente trabajo se caracterizaron estándares, normas y modelos para la realización de pruebas de software. Se evaluaron Planes de Pruebas de proyectos similares, herramientas y recursos para la realización de las pruebas en el proyecto SAGEB. Se definieron cómo serán evaluados los riesgos que puedan surgir en el proceso de pruebas y la estrategia para la realización de las pruebas de software en el proyecto SAGEB. Además se elaboro el Plan de Pruebas para el Proyecto SAGEB.

Los objetivos propuestos para este trabajo fueron plenamente cumplidos a través de las actividades realizadas, la estrategia propuesta está siendo implantada actualmente en el proyecto SAGEB, arrojando resultados satisfactorios en el mismo.

Recomendaciones:

- Profundizar en el estudio y aplicación de las estrategias de pruebas.
- Aplicar la estrategia expuesta en este documento en los proyectos productivos, pues la misma fue creada de manera flexible a cambios, lo que permite que pueda ser adaptada a proyectos con características similares a SAGEB.

Referencias

[1]Pressman, Roger S. Ingeniería de software un enfoque práctico, 2002 [Citado 22 de enero2010]. http://eva.uci.cu/mod/resource/view.php?id=21621&subdir=/Ingenieria_del_Software_un_enfoque_práctico

[2] Jacobson, I.; Booch, G. y Rumbaugh, J.; "El Proceso Unificado de Desarrollo de software". 2000. Addison-Wesley.
http://eva.uci.cu/mod/resource/view.php?id=21621&subdir=/El_proceso_unificado_de_desarrollo_de_software

[3] IBM(R) Rational Unified Process(R) © Copyright IBM Corp. 1987, 2006.

[4] **Almenares, Liudmila Sánchez.** *Cómo realizar pruebas de carga y estres en JMeter.* Ciudad Habana : s.n., 2008.

[5] Diccionario de la Lengua española (Vigésima segunda edición) http://www.rae.es/rae.html

[6]Guzmán Arenas, Adolfo. 2003. *Mitos, creencias y supersticiones sobre la calidad del software y de su enseñanza.* 2003.

Sitios consultados:

Sofonic. [En línea] http://shadow-security-scanner.softonic.com.

http://www.als-es.com/home.php?location=herramientas/entorno-pruebas

Parasoft. [En línea] http://www.parasoft.com/jsp/products.jsp?itemId=

Base de datos: Conjunto de registros (unidades de información relevante) ordenados y clasificados para su posterior consulta, actualización o cualquier tarea de mantenimiento mediante aplicaciones específicas.

Bucles: En programación se entiende por bucle una secuencia de instrucciones que se repite varias veces. Pero las instrucciones sólo se introducen una vez en el código del programa. El número de repeticiones depende del valor de una variable que se llama contador. Cada vez que se ejecuta el código, el contador aumenta su valor hasta alcanzar un valor determinado en el que se detiene el bucle.

Caso de Prueba: Un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular.

Ciclo de vida: Periodo de tiempo que comienza con la concepción del producto de software y termina cuando el producto está disponible para su uso. Normalmente, el ciclo de vida del software incluye las fases de concepto, requisitos, diseño, implementación, prueba, instalación, verificación, validación, operación y mantenimiento, y, en ocasiones, retirada. Nota: Esta fases pueden superponerse o realizarse iterativamente.

Clases: Las Clases son como plantillas o modelos que describen como se construyen ciertos tipos de Objeto. Cada vez que se construye un Objeto de una Clase, se crea una instancia de esa Clase ("instance"). Una Clase es una colección de Objetos similares y un Objeto es una instancia de una Clase. Se puede definir una Clase como un modelo que se utiliza para describir uno o más Objetos del mismo tipo.

Componente: Parte discreta de un sistema capaz de operar independientemente, pero diseñada, construida y operada como parte integral del sistema.

Confiabilidad: Posibilidad que tiene un sistema de realizar las funciones para las que fue diseñado sin fallos.

Desarrollador: Un profesional técnico experto del software que está implicado en la definición, el diseño, la puesta en práctica, la evaluación, y el mantenimiento de un sistema de software.

Eficiencia: Medida del grado en que una actividad alcanza sus objetivos, optimizando el uso de los recursos disponibles.

Especificación de requisitos de software: Documentación de requisitos fundamentales (necesarios, esenciales e indispensables) de funcionalidades, rendimiento, restricciones y atributos del software, y sus interfaces externas. Su acrónimo inglés es SRS.

Funcionalidad: Coherencia entre las necesidades detectadas y los resultados que se obtienen con el uso del material.

Hardware: Conjunto de componentes físicos de una computadora. Refiérase a objetos tangibles y palpables como son los discos, lectores de discos, monitores, teclados, las impresoras, tarjetas y chips.

Herramienta de prueba: Sistemas de software y/o de hardware, u otros instrumentos, que se utilizan para medir y evaluar un artefacto del software

Informática: Se refiere al conjunto de técnicas destinadas al tratamiento lógico y automático de la información, con el fin de obtener una mejor toma de decisiones.

Interface: Conexión entre dos componentes de hardware, entre dos aplicaciones o entre un usuario y una aplicación.

Interfaz de usuario: Interfaz que permite la comunicación entre un usuario y un sistema, o los componentes de un sistema.

Interfaz: Una interfaz es la parte de un programa informático que permite a éste comunicarse con el usuario o con otras aplicaciones permitiendo el flujo de información.

Módulo: Un módulo es un componente autocontrolado de un sistema, el cual posee una interfaz bien definida hacia otros componentes; algo es modular si es construido de manera tal que se facilite su ensamblaje, acomodamiento flexible y reparación de sus componentes.

Plugin: Un plugin (o plug-in) es un programa de ordenador que interactúa con otro programa para aportarle una función o utilidad específica, generalmente muy específica. Los plugins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, codificar/decodificar emails, filtrar imágenes de programas gráficos.

Procedimiento: Es la secuencia de acciones concatenadas entre sí, que ordenadas en forma lógica permite cumplir un fin u objetivo predeterminado.

Prueba funcional: Prueba que ignora la mecánica interna de un sistema o un componente y centra la atención sólo en las salidas generadas como respuesta a determinadas entradas y condiciones de ejecución.

Pruebas: Una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto.

Recurso: Los medios físicos, humanos, y económicos necesitados apoyar un proceso, una política, un procedimiento, una meta, o un programa; por ejemplo, materiales, herramientas de hardware y de software, documentos de los estándares, miembros del personal, y fondos.

SC: Solicitud de cambio.

Servidor: Un servidor en informática o computación es: *Una computadora que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de un ordenador y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final.

Sistema: Conjunto de procesos, hardware, software, instalaciones y personas necesarios para realizar un trabajo o cumplir un objetivo.

Software: Los programas de ordenador, procedimientos, y opcionalmente la documentación y los datos asociados que forman parte de un sistema.

Stakeholder: El personal implicado en el negocio.

Subs: Subsistema.

Usabilidad: Medida de la facilidad de uso de un producto o servicio, típicamente una aplicación software o un aparato (hardware).

Redmine: Es una herramienta de gestión de proyectos software con interface web, donde se pueden definir los hitos del proyecto y las tareas a realizar para cada uno de estos hitos, junto a los responsables de dichas tareas.

Validación: Confirmación por examen y aporte de evidencias objetivas de que los requisitos particulares para un uso específico previsto han sido satisfechos.

Verificación: La constatación ocular o comprobación mediante muestreo, medición, pruebas de laboratorio, o examen de documentos que se realiza para evaluar la conformidad en un momento determinado.

NC: No conformidad.

Anexo1: Diseño de Casos de Prueba.

Descripción General.

Condiciones de Ejecución.

Secciones a probar en el Caso de Uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
<SC 1: Nombre de la sección>	*<EC 1.1: Nombre del Escenario.>*	*<Descripción de la Funcionalidad.>*
	EC 1.2: Nombre del Escenario.	*<Descripción de la Funcionalidad.>*
	EC 1.n: Nombre del Escenario.	<Descripción de la Funcionalidad.>

Descripción de variable.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]<Se enumera todos los	<Nombre de campo de entrada>	<La clasificación es según el componente de diseño	<Se especifica si el campo puede ser	<Una breve descripción de los datos

Matriz de Datos

SC 1<Nombre de la Sección >

Escenario	Variable 1 *(Nombre de la variable)*	Variable 2 *(Nombre de la variable)*	Variable N *(Nombre de la variable)*	Respuesta del Sistema	Resultado de la Prueba	Flujo Central

[Registro de defectos y dificultades detectados.

Elemento	No Conformidad	Aspecto Correspondiente	Etapa de Detección	Significativa	No Significativa	Estado NC	Respuesta del Equipo de Desarrollo

Anexo2: Diseño de Resumen de prueba

Elementos revisados y resultados parciales obtenidos en la Jornada de Trabajo.

Control de Versiones:

No conformidades detectadas.

Cant. NC significativas	Cant. NC no significativas	Cantidad de recomendaciones	Observaciones

Causas de los Resultados Obtenidos Negativamente

Elemento a revisar	Resultados obtenidos

Resultados obtenidos de cada elemento a revisar	Causas

Condiciones de Trabajo existente.

Recursos materiales	
Condiciones ambientales	
Organización del trabajo	

Anexo3: Diseño de Plan de Pruebas:

Roles y responsabilidades

Rol	Cantidad	Responsabilidad
Nombre del rol	*Cantidad*	*Lista de responsabilidades dentro del proceso de revisión*

Escenario de pruebas.

Despliegue del sistema

Recursos del sistema

Servidores

Recurso	TIpo
[Servidor de datos]	*2 procesadores Intel Xeon, 1 GB RAM*

PC Clientes

PC Clientes	
Cantidad	
Descripción	
Software base	
Servicios	

Requerimientos a probar

Estrategia de pruebas de aceptación

Evaluación de las pruebas

Cronograma

	Tarea	Fecha	Responsable	Participantes	Observaciones
	Nombre de la tarea				

Anexo 4 Plan de prueba para el proyecto SAGEB:

Introducción

El presente documento se encarga de describir los procesos necesarios para ejecutar las pruebas durante el desarrollo del proyecto SAGEB, el cual tiene como propósito reunir toda la información necesaria para planear y controlar el esfuerzo de probar el software, atendiendo a determinados objetivos:

• Identificar los elementos que pueden ser objetivo de pruebas.

• Identificar los recursos requeridos para la realización de las pruebas.

• Mostrar el resultado de las pruebas.

Además se encuentran referenciados los roles y las responsabilidades que llevarán a cabo las pruebas y el escenario donde se realizarán las mismas.

Alcance

Este plan de prueba ha sido elaborado según la plantilla establecida por la DCS de la universidad, teniendo como punto de referencia el propuesto por el Proceso Unificado Racional (RUP), ha sido adaptado a las condiciones exclusivas del proyecto SAGEB. Su alcance radica en que es aplicable al flujo de trabajo planificado para el proyecto mencionado, pero a la vez, pretende entregar pautas generales como futuro modelo inicial para ser usado por cualquier otro proyecto software que desarrolle un producto similar.

Roles y responsabilidades

Roles y responsabilidades El grupo de calidad de SAGEB será conformado por un total de 12 personas, atendiendo a la cantidad de casos de uso existentes. Un líder de grupo de calidad, dos analistas de pruebas, tres diseñadores de prueba y seis probadores.

Rol	Cant	Responsabilidad
Lider de calidad	1	Vela por el cumplimiento del plan de pruebas. Es una persona preparada. Su objetivo principal se centra en asegurarse que la aplicación se ajusta a los requerimientos y mantenerse lo más lejano posible de errores. Provee una metodología para realizar las pruebas. Evalúa los resultados que se obtienen al realizar las pruebas de calidad.
Analista de pruebas	2	Deben tener total dominio de las características del sistema creado en el proyecto SAGEB. Identifican y definen las pruebas requeridas. Monitorean el progreso de las mismas y el resultado en cada ciclo de prueba. Evalúan la calidad total experimentada como un resultado de las actividades de prueba.
Diseñador de pruebas	3	Diseña los casos de pruebas. Valora y documenta el efecto de las pruebas realizadas al producto. Define las listas de chequeo
Probador	6	Realizan las pruebas necesarias y el registro del resultado de las mismas.

Tabla 1: Roles y Responsabilidades

Escenario de pruebas.

Las pruebas se realizarán en el laboratorio de proyecto de SAGEB, situado en el Docente2, lab 302 de la Universidad de las Ciencias Informáticas (UCI). Cuentan con seis PCs cliente un servidor de BD y un servidor de la aplicación.

Recursos del sistema

Servidores

Recurso	Tipo
Servidor de la aplicación (2 uno de explotación y otro de respaldo)	Total de servidores: 2, uno para explotación y el otro de respaldo: Procesador core duo , RAM: 4 HB, Sistema operativo: Windows Server 2003, Máquina virtual de Java: JDK 6.0, Servidor web: Apache Tomcat 6.0, Programas para enviar y recibir mensajes SWIFT y SLBTR.
Servidor de base de datos	Procesador core duo, RAM: 4 HB, Capacidad disco duro: 250 HB, Sistema operativo: Windows Server 2003, Microsoft Sql Server 2005

PC Clientes

PC Clientes	
Cantidad	6
Descripción	Pentium IV, RAM: 256 Mb
Software base	Sistema operativo: Windows OS
Servicios	
Navegador web: Mozilla Firefox 3.5	

Estrategia de pruebas de aceptación

Pruebas de Unidad

Pruebas de Funcionalidad:

Descripción: Estas pruebas se aplican para verificar que el módulo se adecúa a los requerimientos funcionales. Es importante decir que se realizarán a cada uno de los módulos del proyecto, a medida que estos vayan saliendo. ***Procedimiento para realizar la prueba***: Se realizarán pruebas de funcionalidad (explicadas en el capítulo anterior) a cada uno de los 37 módulos de forma independiente, las cuales iniciarán con los juegos de datos proporcionados por el cliente y los casos de pruebas basados en los casos de uso creados por los analistas del proyecto. El diseñador confecciona los casos de prueba y el líder de calidad los revisa. La información generada durante el proceso de prueba será guardada en el expediente de proyecto en Implementación y Prueba. Una vez diseñados y aprobados los casos de prueba el probador los ejecuta y documenta los resultados obtenidos, si se

detecta alguna no conformidad, estas deben ser registradas en la tabla de no conformidades que se encuentra en la plantilla de Diseño de Casos de Prueba y deben ser especificadas en el resumen de prueba

Pruebas de usabilidad

Se realizarán pruebas usabilidad a cada uno de los 37 módulos de forma independiente, las cuales iniciarán con los casos de pruebas diseñados los cuales estarán enfocados principalmente a las características de interfaz de usuarios. El diseñador confecciona los casos de prueba y el líder de calidad los revisa. La información generada durante el proceso de prueba será guardada en el expediente de proyecto en Implementación y Prueba. Una vez diseñados y aprobados los casos de prueba, el probador los ejecuta y documenta los resultados obtenidos, si se detecta alguna no conformidad, estas deben ser registradas en la tabla de no conformidades que se encuentra en la plantilla de Diseño de Casos de Prueba y deben ser especificadas en el resumen de prueba

Pruebas de integración:

En el proyecto SAGEB se utiliza el Subversion que no es más que un software de sistema de control de versiones. El Subversion da la posibilidad de poder acceder al repositorio a través de redes, permitiendo que los desarrolladores puedan ir modificando indistintamente a la misma vez desde cualquier punto de trabajo en los laboratorios del proyecto. Trayendo como ventaja poder progresar rápidamente en la implementación de software y tener el control de los cambios que realicen los desarrolladores. Los desarrolladores una vez que suben los cambios que realizan a los módulos el sistema reconoce cuando hay error a la hora de integrar los módulos. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada.
Por otra parte se realizarán pruebas independientes, las cuales las realizará el arquitecto del proyecto SAGEB, ya que tiene los conocimientos necesarios para validar que la integración de los módulos no tenga ningún defecto para llegar al producto final con los resultados esperados.

Prueba de Validación parcial

Estas pruebas son internas, se realizan por los funcionales de la entidad cliente, en este caso Banco Nacional de Cuba (BNC), en la fase de Elaboración. Estas pruebas estarán divididas en dos bloques (bloque 1: primer 50% terminado (es decir que ya hayan terminado las pruebas de unidad) y bloque dos: el resto de los módulos), la primera se planifica en el momento que se terminen las pruebas de unidad al primer 50% de los módulos terminados y una segunda parte que se realizaría con el resto de los módulos. Los clientes proporcionarán los Juegos de Datos para probar el sistema, estas pruebas dispondrán de 2 Iteraciones por cada bloque. Estas pruebas de validación parcial garantizan que el cliente vaya monitoreando el estado del software, pero además son un medidor importante antes de realizar las pruebas de Aceptación y el despliegue del proyecto porque permite ver y corregir errores antes que el producto este terminado completamente.

Pruebas de Sistema

Prueba de Seguridad:***Descripción***: Se verifican los mecanismos de protección incorporados en el sistema. ***Procedimiento para realizar la prueba***: Las pruebas de seguridad están enfocadas hacia la seguridad de la red del sistema. Para la ejecución de dichas pruebas se propone el uso de la herramienta automatizada Shadow Security Scanner

Prueba de Resistencia:***Descripción***: Se realizan pruebas a fallos de sistema. ***Procedimiento para realizar la prueba***: Existe una gran similitud entre las pruebas de resistencia y las pruebas de rendimiento, se diferencian en los objetivos de cada una. La herramienta automatizada Jmeter

Prueba de Rendimiento ***Descripción***: Está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. ***Procedimiento para realizar la prueba:*** Para la realización de las pruebas de rendimiento se propone la utilización de la herramienta automatizada JMeter

Pruebas de Aceptación:***Descripción***: Es la prueba final antes del despliegue del sistema. **Procedimiento para realizar la prueba:** Los errores y cambios que se soliciten quedan recogidos en las plantilla de no conformidades del expediente de proyecto. En caso de no encontrar errores el cliente deja corroborado la aceptación del producto. Y se debe de validar la posibilidad de cambios.

Cronograma

N	Tarea	Fecha	Responsable	Participantes	Observaciones
1	Analizar de las pruebas a realizar.	20/01/2010	Líder de calidad.	Analista de prueba.	
2	*Diseñar casos de Prueba de unidad.*	25/02/2010 al 16/03/2010	Líder de calidad.	Diseñadores de prueba.	Se realizarán durante la fase de elaboración a la par del proceso de implementación.
3	*Capacitar.*	25/02/2010 al 16/03/2010	Líder de calidad.	Probadores.	Se realizara a la par del proceso de Diseño.
4	*Ejecutar pruebas de unidad.*	20/03/2010	Diseñador de prueba.	Probadores.	Se tienen en cuanta el periodo de Iteraciones previstas para estas pruebas.
5	*Ejecutar pruebas de Integración.*	Durante la implementación	Arquitecto	Desarrolladores y Arquitecto.	Son realizadas en la fase de elaboración.
6	*Ejecutar pruebas de Validación Parcial.*	01/04/2010	Líder de calidad.	Cliente.	Consta de dos iteraciones de 1 semana cada una.
7	Analizar de las pruebas a realizar.	20/04/2010	Líder de calidad.	Analista de prueba.	
8	*Diseñar casos de Prueba de Sistema.*	5/ 05/2010	Líder de calidad.	Diseñadores de prueba.	Los casos de prueba deben exhaustivos para todas las pruebas que se realizan en el nivel de Sistema.
9	*Ejecutar prueba de Sistema.*	10/05/2010.	Analista y Diseñador de pruebas.	Probador.	Se tienen en cuanta el periodo de Iteraciones previstas para estas pruebas.
10.	*Ejecutar prueba de Aceptación.*	05/06/2010	Diseñado.	Cliente y Analista.	Se tienen en cuanta el periodo de Iteraciones previstas para estas

					pruebas.

Anexo 5: Ficha técnica Shadow Security Scanner

Nombre	Shadow Security Scanner		**Versión**	7.303
Sistemas Operativos	Win98, 98SE, Me, 2000, NT, XP		**Licencia** Libre	
Tipos de pruebas	Es capaz de analizar un rango de IP`s, y luego de hacerlo, brinda un informe en el cuál se incluyen los puertos abiertos, agujeros de seguridad, shares compartidos y demás.		**Código** Abierto	
Funcionalidades que aporta el Software	Es una poderosa herramienta de control de la red, que analiza el sistema íntegramente en busca de errores. Puede ejecutar más de 4.000 tipos de auditorías diferentes Lo análisis son muy rápidos y se ofrecen informes de cada una de las vulnerabilidades y sus posibles soluciones, además de actualizarse automáticamente a través de Internet.			
Características técnicas	Memoria RAM	Por lo menos 512MB de RAM.		
	Plataformas	Windows, UNIX, Linux, Solaris, HP, CISCO, FreeBSD, OpenBSD y Net BSD.		
	Servidor Web	FTP, SSH, Telnet, SMTP, DNS, Finger, HTTP, POP3, IMAP, IRC, Windows Media Service, Terminal Service, NetBIOS, NFS, NNTP, SNMP, Squid, LDAP, HTTPS, SSL, TCP/IP, UDP y servicios del Registro		
Precio herramienta	gratis	**Precio Licencia / Tiempo de uso**	Versión de prueba 15 dias.	
URL Descarga	http://descargar.mp3.es/lv/software/download			

Anexo 6: Ficha técnica JMeter

Nombre	JMeter		**Versión**	2.3.3
Patrocinador	Apache Jakarta Projects			
Descripción patrocinador	The Apache Software Foundation (ASF) es una organización sin fines de lucro (clasificado como 501 (c) (3) en los Estados Unidos) para apoyar proyectos de software de Apache, incluido el servidor HTTP Apache.			
URL Patrocinador	http://jakarta.apache.org/jmeter/			
Sistemas Operativos	Windows [XP,Window 7], Linux [Debian, Ubuntu]		**Licencia** Libre	
Tipos de pruebas	Rendimiento, estrés.		**Código** Abierto	
Funcionalidades que aporta el Software	Puede cargar y probar el rendimiento del servidor muchos tipos diferentes: Web - HTTP, HTTPS, SOAP, base de datos o por medio de JDBC, LDAP, JMS Correo - POP3 (S) e IMAP (S) Completa portabilidad y el 100% de pureza de Java. El marco de muestreo permite multithreading completa concurrentes por parte de muchos hilos y muestreo simultáneo de diferentes funciones por grupos de hilos separados.			
Características técnicas	Memoria RAM	512		
	Tipo de PC	Asus, Intel,Hanel		

	Plataformas	Java	
	SGBD	PostgreSQL, MySQL, Oracle, SQL Server	
	Navegador	Internet Explorer [6.x, 7.x], Mozilla Firefox [2.x,3.x],	
	Servidor Web	Apache, Tomcat	
Precio herramienta	Gratis	**Precio Licencia / Tiempo de uso**	Libre
URL Descarga	http://apache-jmeter.softonic.com/descargar		