UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
Facultad 9

Sistema de Gestión de Datos Geológicos. Módulo: Inventario de Aguas Minerales. Rol Diseñador de Casos de Prueba.

Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas

Autor:

Yailet Cano Gómez

Tutor:

Ing. Rocny Morales Delgado

Cuidad de La Habana, julio 2010

Año 52 de la Revolución.

"…La lucha por la calidad del producto es una lucha revolucionaria y de vanguardia..."

Ernesto Che Guevara

DEDICATORIA

A la memoria de mi abuelito Oscar por haber sido como mi padre mientras la vida le dio la posibilidad de estar a mi lado, queriéndome, mimándome y consintiéndome como sólo él supo hacerlo. Sé que aunque hoy no pueda estar aquí conmigo, desde el cielo me está mirando y sintiéndose muy orgulloso de mí. A mi padrecito y a mi madrecita porque ustedes son la luz que me guía cuando pienso que todo está perdido, que el mundo se me cae encima, por haberme traído a la tierra y brindarme esta vida plena, sin sacrificios y llena de comodidades. Gracias por la educación humilde pero sincera que siempre me han dado, y que será el ejemplo de la educación que trataré de darles a mis futuros hijos. A ustedes con todo mi amor les regalo el presente y futuro que una vez soñaron para mí. A mi hermana por dedicarme sus mejores sentimientos, por ser mi orgullo y de quién espero ser ejemplo en sus estudios, quién a pesar de ser mi hermanita menor siempre me apoya y sabe darme sabios consejos. A mis abuelos por su amor infinito, por mimarme y malcriarme toda mi vida, por siempre tener tiempo para mí, por ser esos viejitos lindos que yo adoro y quisiera tener a mi lado siempre. A mi tía Maricela por ser esa persona intachable que como una madre, nunca ha escatimado esfuerzos para ayudarme, por brindarme su hombro como medio de seguridad y confianza. Porque sé, sin lugar a dudas lo mucho que me quiere. A mi primo Oscarito por ser tan bueno y especial conmigo, por inspirarme a seguir adelante, por su constante, apoyo, amor y confianza. A mis ahijados por su inocencia y su gracia. Por ser los niños de mi vida, por conquistar mi corazón con sus sonrisas y travesuras. A toda mi familia por su amor y dedicación. A Darly por ser el mejor de mis amigos. El amigo de toda una vida.

AGRADECIMIENTOS

El trabajo de diploma representa la culminación de los estudios pertenecientes a la educación superior, en el que todo joven logra adquirir los conocimientos necesarios para su vida profesional. Es un sueño, que se ha convertido tras cinco años de sacrificio, abnegación y responsabilidad en una realidad perceptible. Son muchas las personas que de una u otra forma han contribuido a convertir mis sueños en realidad, y aunque se me hace casi imposible mencionarlos a todos, quiero agradecerles en especial y de todo corazón:

A mis padres porque sin lugar a dudas son lo mejor que me ha dado la vida, por ser mi ejemplo y mi guía, por su dedicación y confianza en mí, por educarme con principio, responsabilidad y mucho amor, por quererme como sólo ustedes saben hacerlo, por siempre resolver todos mis problemas y hacerlo con sabiduría y mucho amor, por ser los mejores padres del mundo, este triunfo es de ustedes, y me siento orgullosa de ser su hija.

A Yani mi única hermanita y lo mejor que me ha dado mi mamá, gracias por adorarme, por tu confianza sin límites, por ser tan linda conmigo, por tener ese corazoncito tan grande, por todo el amor que te tengo. Porque sin ti, mi niña la vida no tendría sentido, me siento muy orgullosa de poder llamarte hermana.

A mi tía Maricela, que es tan importante y especial en mi vida, por ser como una madre para mí, por ayudarme, por estar siempre conmigo desde que era una bebita, por mimarme y consentirme siempre, por ser mi tía adorada.

A mis abuelos Lola, Mima y Pipo, por lo mucho que me quieren, por la comida tan rica que hacen mis abuelas, por ayudarme siempre que lo he necesitado, por ser los padres de la familia, sin ustedes la felicidad no sería completa.

A mi primo Oscarito por lo jodedor que es pero a la vez, por lo bueno y lindo que es conmigo, por darme siempre su apoyo y confiar siempre en mí, por lo mucho, mucho que te quiero.

AGRADECIMIENTOS

A mis tíos y tías que siempre me han acompañado y siempre han sido muy importantes en mi vida, por todo el apoyo, la comprensión y la confianza que siempre he recibido de ustedes: Mingi, Martha, Wilber, Maritza, Mirelda, Juanito, Maritza, Rafael, y Pepito. A todos los quiero mucho.

A mis primos y primas por todos los momentos que hemos pasado juntos por lo mucho que me quieren y el apoyo que siempre he recibido de ustedes, Santi, Yoa, Lianesita, Ilber, Yoe y Noraisis.

A mi Tutor Rocny por su ayuda en el desarrollo de la tesis y su preocupación por mis resultados.

A mi amigo Alexei por estar siempre de mi lado aconsejándome, comprendiéndome, soportando mis malacrianzas, por todo su cariño, su amistad incondicional y la confianza que siempre ha tenido en mí, por todo lo que me ha enseñado. Porque hubo momentos en los que me sentí realmente sola, pero él estuvo ahí para decirme tú no estás sola, me tienes a mí.

A mi flaquita Dayami por estar a mi lado cuando más falta me hizo una amiga de verdad, siempre voy recordar con cariño los muchos momentos que compartimos.

A las chicas que conocí cuando empezaba la universidad y más tarde consideré amigas, porque aprendí a confiar en ellas y de las que recibí apoyo y cariño: las fantásticas Esther, La china y Sunamy son muy buenas amigas al aguantar mis locuras y ataques de nervios, las quiero mucho.

A mis amigos Mariano, Fale, Osmeidis e Isnán por su cariño, por su confianza, por aguantar mis locuras y permanecer siempre a mi lado, por lo mucho que los quiero.

A Grisel y Mauricio porque de alguna forma también han contribuido en la realización de este sueño.

A diosito por ponerme en un hogar maravilloso al nacer, y darme la posibilidad de crecer rodeada del amor, el cariño, el apoyo y la comprensión de todos ustedes, porque sin mis padres, mi hermana, mis abuelos, mis tíos, mis primos y mis amigos la vida sería simple monotonía.

Yailet

Resumen

En la actualidad las empresas dedicadas a la producción de software buscan diversas vías para lograr una mayor eficiencia en el desarrollo de su producto. La calidad del software es un tema que ha cobrado gran importancia dentro del mercado internacional debido a las exigencias de los clientes; por lo que lograr una mejor producción y la calidad del producto es el objetivo principal de la industria del software. En Cuba se encuentran cada día más organizaciones sumergidas en este proceso de automatización, con lo que se trata de lograr en gran medida un aumento en la producción y la calidad del software.

Entre los proyectos que tienen lugar en la Universidad de las Ciencias Informáticas se encuentra el Sistema de Gestión de Datos Geológicos el cual es realizado para la Oficina Nacional de Recursos Minerales. Dada la importancia de este sistema y la necesidad de entregar el producto libre de errores se le aplicaron diferentes pruebas a todos los módulos del proyecto. El presente trabajo se enfoca en la realización de las pruebas de software a uno de sus módulos, el Módulo Inventario de Aguas Minerales, con lo que se pretende lograr en gran medida el correcto funcionamiento de dicho módulo.

PALABRAS CLAVES

Pruebas de Software, Producto, Calidad

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente con el mismo.

Para que así conste firmo la presente a los____ días del mes de _____ del año_____.

Autora: Yailet Cano Gómez

_____

Tutor: Rocny Morales Delgado

_____

Índice de Tabla

Tabla 1: Diseño de Caso de Prueba de Gestionar Yacimiento. 29

Tabla 2: Diseño de Caso de Prueba de Gestionar Fuente de Agua. 32

Tabla 3: Diseño de Caso de Prueba de Gestionar Recurso Disponible. 35

Tabla 4: Resultados de Gestionar Yacimiento. 49

Tabla 5: Resultado de Gestionar Fuente de Agua. 50

Tabla 6: Resultado de Gestionar Recurso Disponible. 51

Tabla 7: Resultado de Gestionar Recurso de Explotación. 53

Tabla 8: Resultado de Gestionar Recurso en Explotación. 53

Tabla 9: Resultado de Gestionar Control Externo. 54

Índice de Figuras

Figura 1: Ciclo de desarrollo según RUP 8

Figura 2: Método de Prueba Caja Negra 18

Figura 3: Grafo de Flujo ó Grafo del Programa 20

Figura 4: Método de Prueba de Caja Negra 22

Figura 5: Grafo de Flujo del Método Buscar Fuente de Agua. 45

Figura 6: Grafo de Flujo del Método Eliminar Recurso Disponible 46

Figura 7: Grafo de Flujo del Método Buscar Yacimiento 41

Índice

Introducción 1

Capítulo 1: Fundamentación Teórica 4

1.1 Introducción 4

1.2 Conceptos y definiciones 4
1.2.1 Calidad de software 4
1.2.2 Prueba de Software 5
1.2.3 Estrategias de prueba 6

1.3 Metodologías de Desarrollo de Software 6

1.4 Proceso Unificado de Desarrollo de Software (RUP) 7
1.4.1 Características fundamentales del ciclo de vida en RUP 9

1.5 El Proceso de Pruebas en el Ciclo de Desarrollo. Objetivos, Trabajadores, Actividades y Artefactos según la Metodología RUP 9
1.5.1 Objetivos 10
1.5.2 Trabajadores y Actividades que establece RUP para el flujo de trabajo de prueba 10
1.5.3 Artefactos del flujo de trabajo de Prueba 11

1.6 Rol: Diseñador de Casos de Prueba 12
1.6.1 Tareas que realiza el Diseñador de casos de prueba 13

1.7 Niveles de prueba 13
1.7.1 Pruebas de Unidad 13
1.7.2 Pruebas de integración 14
1.7.3 Prueba de Validación 15
1.7.4 Prueba del sistema 15
1.7.5 Prueba de Aceptación 17

1.8 Métodos de prueba 18
1.8.1 Método de Prueba de Caja Blanca 18
1.8.2 Método de prueba de caja negra 22

1.9 Conclusiones 24

Capítulo 2: Diseño y Aplicación de las Pruebas 25

2.1 Introducción 25

2.2 Descripción del Sistema a Probar 25

ÍNDICE DE CONTENIDO

2.3 Características a probar 25
2.4 Plan de Pruebas 26
2.4.1 Objetivos 27
2.5 Estrategias de pruebas 28
2.6 Pasos a seguir en las pruebas 28
2.7 Diseño de Casos de Prueba de Caja Negra 29
2.8 Diseño de Pruebas de Caja Blanca. Técnica, Camino Básico. 38
Caso de Prueba del Caso de Uso Gestionar Yacimiento. 38
2.9 Conclusiones 47
Capítulo 3: Análisis de los resultados Obtenidos 48
3.1 Introducción 48
3.2 Resultados de las pruebas a la documentación del módulo Inventario de Aguas Minerales 48
3.3 Resultados obtenidos de aplicar las pruebas de Caja Blanca y Caja Negra al módulo Inventario de Aguas Minerales 49
3.3.1 Funcionalidad: Gestionar Yacimiento 49
3.3.2 Funcionalidad: Gestionar Fuente de Agua 50
3.3.3 Funcionalidad: Gestionar Recurso Disponible 51
3.3.4 Funcionalidad: Gestionar Recurso de Explotación 53
3.3.5 Funcionalidad: Gestionar Recurso en Explotación 53
3.3.6 Funcionalidad: Gestionar Control Externo. 54
3.4 Evaluación del módulo Inventario de Aguas Minerales 56
3.5 Conclusiones 57
Conclusiones 58
Recomendaciones 59
Referencias Bibliográficas 60
Anexos 63
Glosario de Términos 67

Introducción

El desarrollo que han alcanzado las tecnologías en las últimas décadas ha marcado un cambio notable en el manejo de la información en las distintas ramas de la sociedad. Cambio que ha permitido a las empresas y organizaciones que manejan gran cantidad de información lograr un mejor acceso, almacenamiento y utilización de los datos que procesan. De ahí la necesidad de automatizar sus actividades, logrando que el procesamiento de la información que manejan a diario sea más seguro y fácil. El desarrollo de este proceso de automatización implica la realización de una serie de actividades donde existe una alta probabilidad de cometer errores por parte del equipo de desarrollo. La calidad del producto es un tema de interés a nivel internacional. Su importancia está dada por las exigencias del cliente que se hacen cada vez mayores y la competencia entre las empresas por lograr un mejor producto.

Uno de los pilares fundamentales que sustentan el desarrollo de software en Cuba lo constituye la Universidad de las Ciencias Informáticas. En este centro se desarrolla una parte considerable del software nacional y se llevan a cabo diversos proyectos productivos para diferentes empresas tanto nacionales como extranjeras. Por tanto va encaminada a convertirse en una industria donde prevalece un creciente aumento y desarrollo de la producción de software. La universidad cuenta con 10 facultades entre las que se encuentra la facultad 9.

Esta facultad tiene bajo su responsabilidad el desarrollo de varios proyectos productivos, entre ellos el proyecto Sistema de Gestión de Datos Geológicos (SGDG). El proyecto cuenta con diferentes módulos: Búsqueda Referativa, Inventario de Minerales Sólidos, Registro Petrolero, Inventario de Petróleo y Gas, Registro Minero e Inventario de Aguas Minerales.

El desarrollo de este último debe contar con una serie de funcionalidades que permitan administrar y proveer toda la información sobre el estado de los recursos y aguas minerales; controlando y garantizando el uso racional de las mismas, así como gestionar la información referida a los yacimientos, fuentes de agua y recursos disponibles. Además, debe proporcionar un inventario anual del estado de los recursos disponibles, recursos en y de explotación y los controles externos de las aguas minerales del país. El sistema a desarrollar es muy complejo, para su construcción se cuenta con un equipo de desarrollo con la responsabilidad de entregar el producto en tiempo al cliente. Como todo proceso de desarrollo está expenso a que se cometan errores en cualquiera de las etapas de construcción del sistema, y dada la

importancia de dicho software se hace necesario realizar una serie de pruebas para detectar las deficiencias y garantizar que el software sea entregado libre de defectos.

A raíz de lo anteriormente planteado y de la necesidad de contar con un producto de alta calidad surge el siguiente **problema a resolver:** ¿cómo lograr la entrega del módulo Inventario de Aguas Minerales libre de defectos? Dicha problemática permite establecer como **objeto de estudio:** el proceso de pruebas de Sistemas de Gestión de Información. Enmarcando el **campo de acción** en: el Diseño e implementación de las pruebas del módulo Inventario de Aguas Minerales. Estableciéndose como **Objetivo General:** elaborar la documentación técnica del proceso de pruebas correspondiente al módulo Inventario de Aguas Minerales que forma parte del Sistema de Gestión de Datos Geológicos. Se establece como **Idea a defender:** el diseño y la implementación de las pruebas al módulo Inventario de Aguas Minerales que forma parte del Sistema de Gestión de Datos Geológicos garantizarán que el mismo se entregue libre de defectos.

Para dar solución al objetivo general se establecieron las siguientes **tareas de la investigación:**

1. Caracterizar el proceso de pruebas según la metodología seleccionada.
2. Caracterizar el rol Diseñador de Casos de Prueba según la metodología seleccionada.
3. Elaborar el plan de pruebas para el módulo Inventario de Aguas Minerales.
4. Diseñar los Casos de Prueba.
5. Realizar las pruebas de Caja Blanca al módulo Inventario de Aguas Minerales.
6. Realizar las pruebas de Caja Negra al módulo Inventario de Aguas Minerales.
7. Caracterizar el estado de la implementación una vez finalizado el proceso de pruebas.
8. Validar las pruebas realizadas.

Para la realización de esta investigación fueron utilizados diferentes **métodos científicos**: Teóricos y Empíricos, estos métodos permiten analizar y sintetizar la información referente al objeto de estudio.

Métodos Teóricos: Estos métodos permiten mostrar las relaciones del objeto de estudio. Son fundamentales para la comprensión de los hechos y para la formulación de la hipótesis de investigación.

- ✓ **Método Histórico-lógico:** Mediante este método se lleva a cabo un estudio sobre los antecedentes de las pruebas de software, y partiendo de la información obtenida arribar a una propuesta de solución que permita asegurar la calidad del producto.
- ✓ **Método analítico-sintético:** Este método es utilizado para dividir la información obtenida del objeto de estudio en partes, y de esta forma poder analizar mejor sus características.

Métodos Empíricos: Estos métodos permiten efectuar el análisis preliminar de la información, así como verificar y comprobar las concepciones teóricas. Su contenido procede fundamentalmente de la experiencia.

- ✓ **Observación:** Este método es utilizado durante toda la investigación, pues tiene en cuenta el criterio y las definiciones de otros autores acerca del objeto de estudio. Además de ser utilizado también para observar el código a la hora de realizar las pruebas.

El presente trabajo de diploma consta de 3 capítulos los que quedan estructurados de la siguiente forma.

Capítulo 1: **Fundamentación Teórica**

En este capítulo se exponen los temas que sustentan la investigación científica, conceptos asociados al problema que permitan un mayor entendimiento del mismo, y se caracteriza el proceso de Prueba según la metodología utilizada en el proyecto.

Capítulo 2: Diseño y aplicación de las pruebas

En este capítulo se plantea el Plan de Pruebas que será aplicado al módulo Inventario de Aguas Minerales del proyecto SGDG, se define la estrategia de pruebas a seguir, se hace referencia a los procedimientos de pruebas que serán el punto de partida para el diseño y la ejecución de los Casos de Pruebas.

Capítulo 3: Análisis de los resultados

En este capítulo se analizan los resultados obtenidos luego de aplicar las pruebas y se elabora un resumen con los principales errores detectados durante el proceso.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se realiza un estudio acerca de las pruebas de software. Se explica de manera precisa los conceptos más importantes que serán necesarios para entender los diferentes temas asociados al dominio del problema. Se profundiza en los temas que sustentan la investigación científica, y se realiza una caracterización de las pruebas de software según la metodología utilizada.

1.2 Conceptos y definiciones

1.2.1 Calidad de software

Es la 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. (Lovelle, 1999)

Es el conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas. (Lovelle, 1999)

Es el desarrollo de software basado en estándares con la funcionalidad y rendimiento total que satisfacen los requerimientos del cliente. (Soto)

Es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario. (Hugo)

Según la Organización Internacional de Estándares (ISO/IEC DEC 9126) la calidad del software es: La totalidad de características de un producto de software que tiene como habilidad, satisfacer las necesidades implícitas o explícitas. Se refiere a los factores de un producto de software que contribuyen a la satisfacción completa y total de las necesidades de un usuario u organización.

La calidad del software constituye una vía para garantizar el cumplimiento de los objetivos que se pretenden lograr con la realización del software. Permite conocer además, el grado con el que el producto cumple con las necesidades y los requerimientos establecidos por el cliente. Los requisitos son la base para lograr la calidad del software. Para alcanzar la calidad suficiente del producto es necesario comprender las necesidades reales de los usuarios. La calidad del software es todavía un logro importante y alcanzarla es un gran objetivo.

1.2.2 Prueba de Software

Es la ejecución de un programa con la intención de descubrir un error. Constituye una técnica experimental para la búsqueda de errores en los programas. (2009)

Son técnicas realizadas para evaluar un determinado producto de software y detectar errores en la aplicación durante su ejecución. (Pressman, (1998))

La prueba de software es la búsqueda constante de errores y fallas que surjan durante el ciclo de desarrollo. (Pressman, (1998))

Es una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente. (Pressman, 2005)

Cualquier intento de demostrar que el software tiene propiedades por debajo de la calidad requerida. (Brito, 2003)

Las Pruebas de Software tienen la función de velar porque se cumplan los requerimientos del sistema y garantizan la calidad del producto final. La prueba será exitosa cuando encuentre errores en la aplicación por lo que constituyen una parte fundamental en el ciclo de desarrollo de un producto. Las pruebas no pueden asegurar que el software no contenga errores, sólo pueden demostrar la existencia de estos. Permiten validar si el software satisface las expectativas de los clientes garantizando el cumplimiento de los requisitos establecidos en la etapa de inicio.

1.2.3 Estrategias de prueba

Las estrategias de prueba son las líneas guía del equipo de prueba de un proyecto de desarrollo de software. (Pressman, (1998))

La Estrategia de Prueba se encarga de velar porque el producto de software que se está construyendo, reúna los requerimientos de la lógica del negocio que el cliente ha pedido realizar mediante el debido contrato de desarrollo de software. (2009)

La Estrategia de Prueba agrupa los métodos de diseño de Casos de Pruebas del software en una serie bien planeada de pasos que terminará en la construcción del software. Además, proporciona un mapa que describe los pasos que se darán como parte de la Prueba. Indica cuándo se planifican y se llevan a cabo estos pasos. Cuánto esfuerzo, tiempo y recursos gastarán. (Pressman, 2005)

La Estrategia de Prueba de Software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba: la planificación, el diseño de Casos de Prueba, la ejecución y los resultados, tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software. (Pressman, 2005)

El proceso de prueba consume gran cantidad de tiempo y esfuerzos, por lo que es necesario crear una buena estrategia para probar el software. De esta forma, se logra disminuir el consumo innecesario tanto de tiempo como de recursos y se detectan mayor cantidad de errores.

1.3 Metodologías de Desarrollo de Software

Cuando se realiza un determinado producto de software, se debe contar con una guía para la correcta organización del trabajo que se desarrolla. Es decir, un patrón que establezca una sucesión organizada de las actividades que se deben desarrollar y las tareas que se deben cumplir.

FUNDAMENTACIÓN TEÓRICA

Las metodologías de desarrollo constituyen la base para la creación de un software, establecen además una serie de pasos organizados en un orden lógico, que deben ser seguidos para lograr los objetivos que se desean alcanzar con la construcción del producto. (Delgado, 2010)

Si el equipo de desarrollo no aplica una determinada metodología que además se ajuste a las necesidades del proyecto, el producto no tendrá la calidad suficiente. Para lograr la calidad de un producto de software se debe tener como premisa la puesta en práctica de una metodología que establezca e integre todas las etapas de desarrollo, y permita controlar de manera eficiente todo este proceso. Permiten además obtener un producto en el tiempo esperado y con el menor costo posible. Existen diversas metodologías, que son aplicadas con el objetivo de alcanzar una mejora continua de todos los procesos relacionados con el desarrollo de un software. No se cuenta con una metodología universal para realizar cualquier proyecto de software, por lo que cada proyecto según sus características debe contar con una metodología para su desarrollo.

1.4 Proceso Unificado de Desarrollo de Software (RUP)

Teniendo en cuenta las características del software que se desea desarrollar se definió en el documento de la arquitectura básica del sistema que la metodología más adecuada para guiar el proceso de desarrollo del Sistema de Gestión de Datos Geológicos es: el Proceso Unificado de Desarrollo de Software (RUP).

RUP es una metodología para la ingeniería de software que va más allá del análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de un producto. El resultado es un proceso basado en componentes, dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. (Delgado, 2010)

RUP tiene como objetivo principal la realización de un producto de alta calidad, en el tiempo establecido entre clientes y desarrolladores, que logre la satisfacción del usuario final. (Ivar Jacobson, 2000)

Como se muestra en la figura 1, RUP agrupa los procesos más importantes en 9 flujos de trabajo, los 6 primeros son de ingeniería y los tres restantes de apoyo. RUP basa el proceso de desarrollo en ciclos

obteniendo un resultado al finalizar cada uno y cada ciclo se realiza mediante fases que concluyen con un hito: (Fernández, 2000)

Inicio (Concepción o conceptualización): En esta fase se modela el negocio, se describe la visión, los objetivos y el alcance del proyecto.

Elaboración: En esta fase la principal función es completar el análisis de los casos de uso y definir la arquitectura del sistema.

Construcción: Esta fase se centra en la elaboración de un producto listo para su utilización, operativo, eficiente, bien documentado y con un manual de usuario.

Transición: En esta fase se propone el producto listo para su instalación a los usuarios finales.

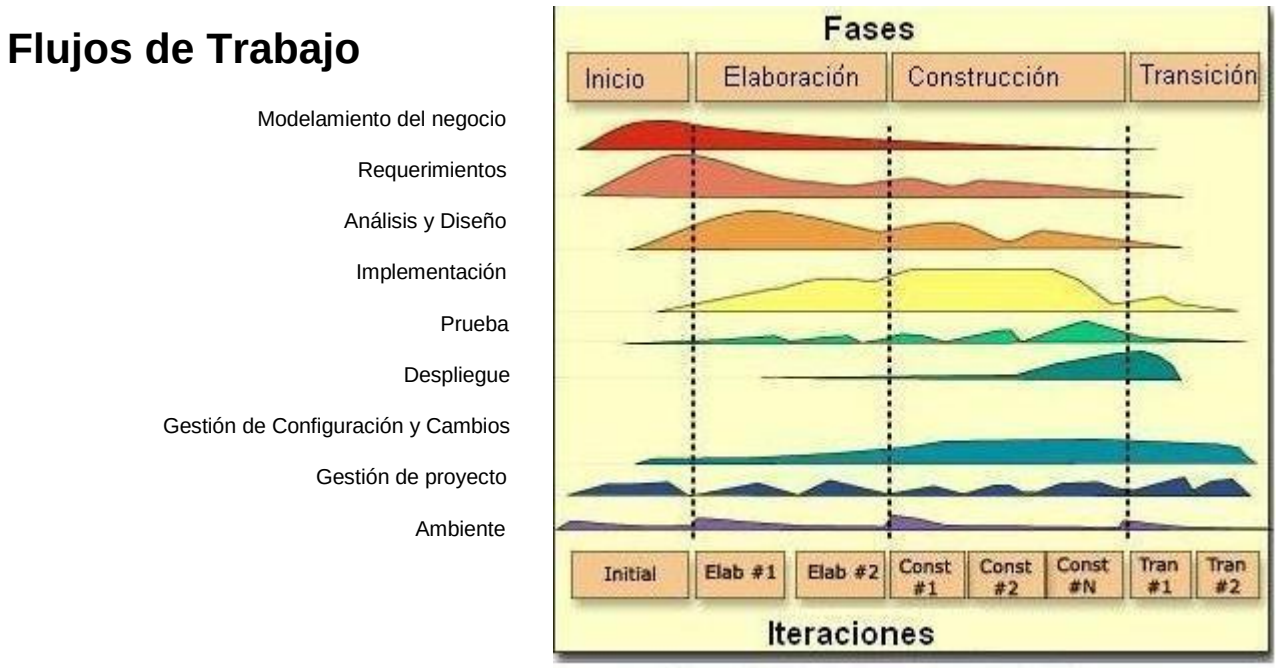

Figura 1: Ciclo de desarrollo según RUP.

1.4.1 Características fundamentales del ciclo de vida en RUP

Dirigido por casos de uso: Los Casos de Uso son el instrumento para validar la arquitectura del software y extraer los Casos de prueba. (Delgado, 2010)

Iterativo e incremental: Durante todo el proceso de desarrollo se producen versiones incrementales que se acercan al producto terminado del producto en desarrollo. (Delgado, 2010)

Centrado en la arquitectura: Los modelos son proyecciones del análisis y el diseño constituye la arquitectura del producto a desarrollar. (Delgado, 2010)

1.5 El Proceso de Pruebas en el Ciclo de Desarrollo. Objetivos, Trabajadores, Actividades y Artefactos según la Metodología RUP

Con el surgimiento de los primeros sistemas informáticos se incluyó a nivel internacional un nuevo proceso en la creación de los mismos: El proceso de prueba. Son muchas y muy variadas las técnicas que se pueden realizar para encontrar defectos en un software y abarcan desde la capacidad que tenga el personal dedicado a detectar fallas en los programas, hasta herramientas automatizadas que facilitan tanto en costo como en tiempo la realización de esta actividad.

¿Por qué Probar?

La necesidad de aplicar pruebas sobre los programas antes de llevarlos al mercado radica en que son enormes y muy constantes las posibilidades de cometer errores por parte de los desarrolladores. Las fallas en un software ocasionan grandes pérdidas económicas, que se tornan difíciles de recuperar. Al realizar una determinada prueba, no sólo se logra una buena calidad y excelencia del producto sino también evitar plazos y presupuestos incumplidos, escasa productividad y pérdida de los clientes. (Pressman, 2005)

1.5.1 Objetivos

Son múltiples los objetivos que se persiguen en la etapa de prueba. El propósito principal es encontrar defectos, es decir validar la corrección durante la ejecución de un programa y de esta forma garantizar la calidad del producto.

Objetivos más importantes de las pruebas

- ✓ Probar si el software no hace lo que debe hacer.
- ✓ Probar si el software hace lo que no debe hacer.
- ✓ Encontrar y documentar los artefactos que puedan afectar la calidad del producto.
- ✓ Validar y probar los requisitos que debe cumplir el software.
- ✓ Validar que los requisitos fueron implementados correctamente.

Objetivos básicos de las pruebas

- ✓ Encontrar errores lo más temprano posible en el ciclo de desarrollo.
- ✓ Asegurar que el producto cumple con los criterios preestablecidos. (Fabian, 2007)

Los objetivos actuales de las pruebas, no sólo tienen que ver con corregir errores, sino con prevenirlos influyendo y controlando el diseño y desarrollo del software.

1.5.2 Trabajadores y Actividades que establece RUP para el flujo de trabajo de prueba

Trabajadores

Administrador de Prueba*:* Es el responsable de que la prueba tenga éxito. Además, involucra defensor de prueba y calidad, planificación y administración de recursos y resolución de problemas que impiden las pruebas. (Pressman, 2005)

Analista de Prueba: Es el responsable de identificar y definir las pruebas requeridas, monitorear el progreso de la prueba y el resultado en cada ciclo. Tiene la responsabilidad de representar

apropiadamente las necesidades de los Stakeholders que no tienen representación regular y directa en el proyecto. (Pressman, 2005)

Diseñador de Prueba: Es el responsable de definir el método de prueba y asegurar su implementación exitosa. Es responsable de identificar las técnicas apropiadas, herramientas e instrucciones para implementar las pruebas necesarias y guiar los recursos correspondientes para las pruebas. (Pressman, 2005)

Probador: Es el responsable durante las actividades principales de las pruebas, es el encargado de conducir las pruebas necesarias y el registro del resultado de las pruebas. (Pressman, 2005)

Actividades

- ✓ Planificar las pruebas.
- ✓ Diseñar las pruebas.
- ✓ Implementar las pruebas.
- ✓ Realizar pruebas de integración.
- ✓ Realizar pruebas de sistema.
- ✓ Evaluar pruebas.

1.5.3 Artefactos del flujo de trabajo de Prueba

Plan de Pruebas

El propósito del plan de pruebas es explicitar el alcance, enfoque, recursos requeridos, calendario, responsables y manejo de riesgos de un proceso de pruebas. (2005) En el Plan de Pruebas realizado para el Módulo Inventario de Aguas Minerales del Proyecto Sistema de Gestión de Datos Geológicos, se identifican los elementos que van a ser probados, así como los recursos necesarios para realizar las Pruebas y La Estrategia de Prueba que se llevará a cabo a la hora de realizar los diseños de casos de prueba.

Un Plan de Pruebas está constituido por un conjunto de pruebas donde cada una de ellas debe dejar bien claro qué tipo de propiedades se quiere probar, por ejemplo: corrección, robustez, fiabilidad, amigabilidad. Dejar claro cómo se mide el resultado. Especificar en qué consiste la prueba (hasta el último detalle de cómo se ejecuta). Definir cuál es el resultado que se espera (identificación, tolerancia.) (Mañas, José A)

Estrategia de Prueba

La Estrategia de Prueba de software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba: la planificación, el diseño de casos de prueba, la ejecución y los resultados, tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software. (2009). La estrategia de prueba constituye líneas guías del equipo de prueba de un proyecto de desarrollo de software.

Procedimiento de Prueba

Especifica cómo realizar uno o varios casos de prueba. Un procedimiento de prueba puede ser una instrucción para un individuo sobre cómo ha de realizar un caso de prueba manualmente o puede ser una especificación de cómo interaccionar manualmente con una herramienta de automatización de pruebas para crear componentes ejecutables de pruebas. (Collazo, 1990)

1.6 Rol: Diseñador de Casos de Prueba

La persona encargada de realizar las pruebas a un software desempeña dentro del equipo de desarrollo el rol: Diseñador de Casos de Prueba. Éste es el responsable de las actividades principales de las pruebas, las cuales incluyen identificar, definir, diseñar, implementar y dirigir las pruebas necesarias. Además de ejecutar las pruebas diseñadas y caracterizar el estado de la implementación una vez finalizado el proceso de pruebas. Debe además verificar los resultados al finalizar cada prueba y realizar un análisis de los resultados obtenidos, en caso de encontrar algún error debe infórmaselo al desarrollador para que este los corrija. (Pressman, 2005)

El Diseñador de Casos de Prueba de un software es responsable de la integridad del modelo de pruebas, asegurando que el modelo cumple con su propósito. Planea las pruebas, selecciona y describe los casos y procedimientos de prueba. El probador es el responsable de los artefactos modelo de prueba, casos de prueba, procedimientos de prueba, evaluación de pruebas y plan de pruebas. (Anna. C Grimán)

1.6.1 Tareas que realiza el Diseñador de casos de prueba

El rol Diseñador de Casos de Prueba es el responsable de realizar las siguientes tareas:

1. Identificar las pruebas que serán realizadas.
2. Implementar pruebas individuales.
3. Preparar y ejecutar las pruebas.
4. Analizar los resultados.
5. Analizar los errores de ejecución.
6. Comunicar los resultados de las pruebas al equipo de desarrollo.

1.7 Niveles de prueba

Teniendo en cuenta que las pruebas son realizadas en determinados momentos del ciclo de vida del software, son agrupadas en niveles de acuerdo con las diferentes etapas del ciclo de desarrollo.

1.7.1 Pruebas de Unidad

La prueba de unidad centra el proceso de verificación en la menor unidad del diseño del software: el componente software o módulo. (Pressman, 2005)

Esta prueba es la de menor escala, está basada en las funciones y procedimientos de los módulos del programa. Una tarea de gran importancia durante esta prueba es la comprobación selectiva de los caminos de ejecución.

Antes de iniciar cualquier otra prueba es necesario probar el flujo de datos de la interfaz del módulo. Si los datos no entran correctamente no tiene sentido aplicar las demás pruebas. Esta prueba también debe comprobar el impacto de los datos globales sobre el módulo. Una prueba unitaria, es una vía para demostrar que los módulos de un programa funcionan correctamente por separado.

1.7.2 Pruebas de integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la integración. El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño. (Pressman, 2005)

Estas pruebas permiten asegurar que los componentes en el modelo de implementación funcionen correctamente cuando son combinados para ejecutar un caso de uso. La prueba de integración es realizada durante la fase de construcción y abarca gran cantidad de módulos.

Una vez que el sistema sea integrado es necesario aplicar las pruebas de integración para asegurar el correcto funcionamiento de los módulos como un todo. Existen diferentes tipos de pruebas de integración que son aplicadas con este propósito.

Prueba de integración descendente

La prueba de integración descendente comienza a integrar los módulos moviéndose hacia abajo, partiendo del módulo de control principal o mayor nivel y luego se incorporan a la estructura los módulos subordinados a él, ya sea mediante el método primero en profundidad o primero en anchura. (Pressman, 2005)

Primero en profundidad: Se mueve primero en forma vertical en la estructura de módulos.

Primero en anchura: Se mueve primero en forma horizontal en la estructura de módulos.

Prueba de integración ascendente

Esta prueba empieza a integrar los módulos partiendo de los niveles más bajos de la estructura del programa. De esta forma, el proceso requerido de los módulos subordinados siempre estará disponible y se elimina la necesidad de resguardarlos.

Prueba de regresión

Luego de aplicar la prueba de integración se añade un nuevo módulo y el software cambia. La prueba de regresión es aplicada como una estrategia de la prueba de integración, esta consiste en ejecutar pruebas que han sido ejecutadas anteriormente para asegurar que los cambios producidos en el software no tienen efectos indeseados.

Prueba de humo

La prueba de humo es muy utilizada cuando se obtiene un producto de software desarrollado en paquetes. Se aplica al sistema de principio a fin, y a pesar de no ser exhaustiva es capaz de detectar errores en la aplicación. Esta prueba además permite simplificar los riesgos de integración y perfeccionar la calidad del producto final.

1.7.3 Prueba de Validación

Una vez realizada la prueba de integración se han encontrado y corregido los errores de interfaz y se puede empezar una serie final de pruebas del software: La prueba de validación. La validación se consigue cuando el software funciona de acuerdo con las expectativas del cliente. (Pressman, 2005)

1.7.4 Pruebas de sistema

Estas pruebas son aplicadas para determinar el correcto funcionamiento del sistema una vez integrados todos los componentes tanto de software como de hardware, en fin la prueba del sistema es ejecutada

con el propósito de asegurar que los elementos del sistema fueron integrados correctamente y que realizan las funciones apropiadas.

Pruebas de recuperación

Es una prueba del sistema que fuerza el fallo del software y verifica que la recuperación del mismo se lleve a cabo correctamente.

- ✓ Si la recuperación es automática hay que evaluar la corrección de la inicialización, de los mecanismos de recuperación del estado del sistema, de la recuperación de los datos y del proceso de arranque.
- ✓ Si la recuperación no es automática, hay que evaluar los tiempos medios de reparación para determinar si están dentro de unos límites aceptables. (Lovelle, 1999)

Prueba de Seguridad

Este tipo de prueba tratará de obtener las claves de acceso al sistema, de esta forma, permite verificar si los mecanismos de protección incorporados al sistema son eficientes. También garantizan el acceso limitado al sistema por parte de los usuarios, estos sólo podrán acceder a los datos que tengan autorización. Su función principal es comprobar la seguridad lógica del sistema, ya que sólo los usuarios con permiso para acceder podrán ejecutar alguna función propia del sistema.

Prueba de Resistencia

Estas pruebas están diseñadas para comprobar el funcionamiento del sistema bajo condiciones atípicas. Ejecuta el sistema de forma que demande recursos en cantidad, como por ejemplo incrementa las frecuencias de datos de entrada, ejecuta casos que requieran el máximo de memoria. El objetivo de aplicarlas es determinar el punto en que el sistema empieza a funcionar por debajo de los requisitos establecidos.

Prueba de Rendimiento

Estas pruebas están diseñadas para probar el rendimiento del software en tiempo de ejecución, dentro de un sistema integrado. Es necesario que estas pruebas comiencen desde el inicio del proyecto, de esta forma, se logra un mayor rendimiento en el sistema. Cuando más se demore en detectar un error de

rendimiento durante el ciclo de desarrollo, mayor será el coste de solución. Por este motivo es recomendable que se inicien en las fases tempranas del desarrollo del proyecto y se amplíen en la fase de construcción.

1.7.5 Prueba de Aceptación

Esta prueba es aplicada para comprobar que el software funciona correctamente según su funcionalidad y rendimiento y que está listo para ser usado por los usuarios finales del sistema, quienes son los encargados de realizar dicha prueba y emitir su aprobación final. Hay desarrolladores que llevan a cabo un proceso denominado alfa o beta. El objetivo que persiguen con esto es descubrir errores que parezcan que sólo el usuario final pueda descubrir.

Prueba Alfa

Es llevada a cabo por un cliente, en el lugar de desarrollo con la presencia del desarrollador como observador del usuario y registrando los errores detectados.

La Prueba Beta

Es realizada por los usuarios finales en el lugar de trabajo de los clientes. Es una prueba realizada en un entorno en la que el desarrollador no puede controlar. El cliente registra los problemas encontrados y se los notifica al desarrollador.

1.8 Métodos de prueba

Existen diferentes métodos de pruebas que son aplicados sobre un determinado producto con el objetivo de detectar y corregir los errores que surjan a medida que avance el proceso de desarrollo del software. Entre los métodos más importantes se encuentran, el método de Prueba de Caja Blanca y el método de Prueba de Caja Negra.

Estas pruebas permiten realizar una prueba completa de todo el sistema, tanto al código como a la interfaz, de esta forma, se obtendrá un producto más confiable, de mayor calidad y que gane prestigio entre los clientes.

1.8.1 Método de Prueba de Caja Blanca.

Las Pruebas de Caja Blanca y como también se les conoce Pruebas de Cristal, es un método de diseño de Casos de Pruebas que usa la estructura de control del diseño procedimental para obtener los Casos de Prueba. Estas se realizan sobre el código fuente a medida que este va ejecutando los casos de prueba.

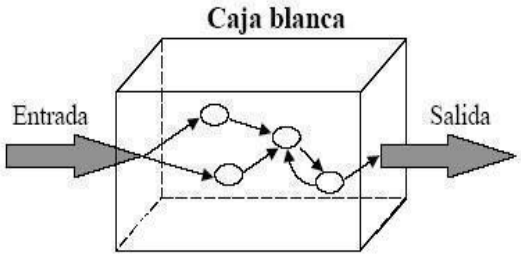

Figura 2: Método de Prueba Caja Blanca.

Mediante este método de pruebas el ingeniero de software puede obtener Casos de Prueba que garanticen: (Brito, 2003)

- ✓ Que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo.
- ✓ Que se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.

- ✓ Que se ejecuten todos los bucles en sus límites y con sus límites operacionales.
- ✓ Que se ejerciten las estructuras internas de datos para asegurar su validez.

La Prueba de Caja Blanca es una de las pruebas de software más importantes, permite lograr una mayor calidad en el producto debido a que logra disminuir en gran medida los errores existen en los programas. Estas pruebas tienen diferentes técnicas. A continuación se relacionan algunas de ellas:

Pruebas de estructura de control

Esta prueba tiene diferentes técnicas como las pruebas del camino básico, prueba de condición, entre otras que, conjuntamente amplían la cobertura de la prueba y mejoran la calidad de la prueba de caja blanca.

Pruebas del camino básico

Es una técnica de prueba de caja blanca que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica del diseño y a partir de ahí elaborar un conjunto básico de caminos de ejecución. A partir este conjunto se derivan casos de pruebas por los cuales va a circular el flujo de control y de esta forma garantizarán que durante las pruebas se ejecuten, por lo menos una vez, cada sentencia del programa. Antes de realizar el método del Camino Básico es necesario representar el flujo de control esto se hace mediante una notación denominada *grafo de flujo* o *grafo del programa*. (Figura 3).Donde cada nodo representa una o más sentencias y las aristas representan el flujo de control.

La técnica del camino básico representa un límite superior para la cantidad de casos de pruebas que deber ser realizados para comprobar que se ejecuta cada camino del programa. Para aplicar esta técnica se deben tener en cuenta los siguientes pasos:

1. Representar el programa en un grafo de flujo.
2. Calcular la complejidad ciclomática.
3. Determinar el conjunto básico de caminos independientes.
4. Derivar los casos de prueba que fuerzan la ejecución de cada camino.

Construcciones estructuradas: secuencia, if, mientras, case.

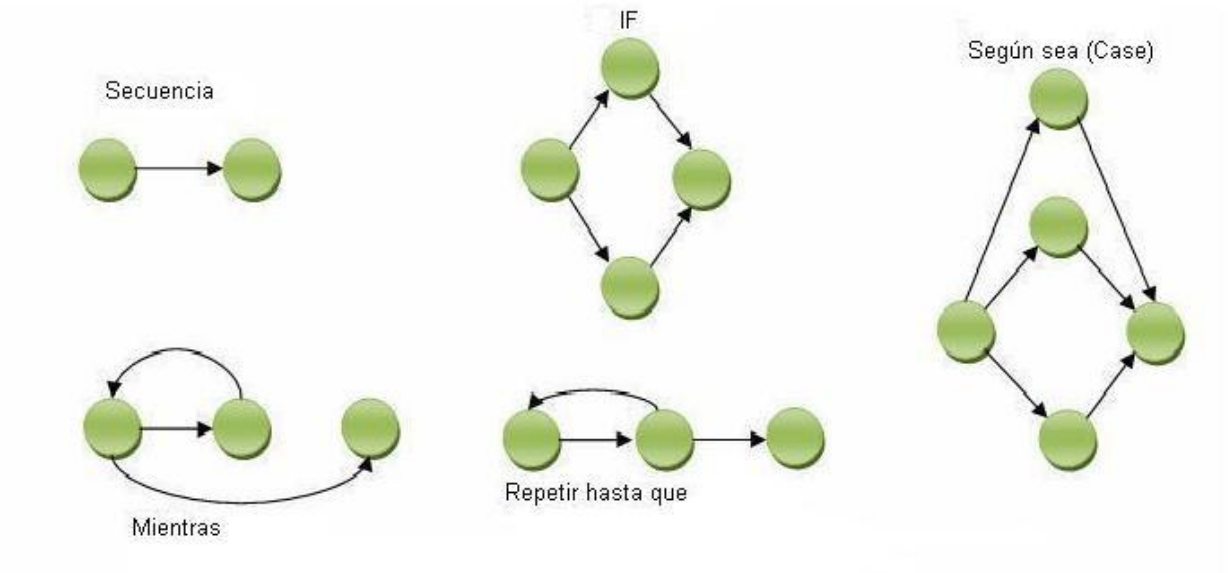

Figura 3: Grafo de Flujo o Grafo del Programa

Complejidad Ciclomática

Según Pressman la complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. [15] Cuando se usa para calcular la complejidad el camino básico, el resultado encontrado define el número de caminos independientes del conjunto básico de un programa. De esta forma, el probador tiene una idea de cuantas pruebas debe realizar para asegurar, al menos una vez, la ejecución de cada sentencia de código.

La complejidad ciclomática puede ser calculada mediante tres formas:

El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

1. Número de regiones = V (G)

2. V(G) = A – N + 2

   A = Número de aristas del grafo.

   N = Número de nodos.

3. V(G) = P + 1

   P = Número de nodos predicados del grafo.

Pruebas de condición

Es un método para el diseño de casos de prueba, con el objetivo de ejercitar las condiciones lógicas contenidas en el módulo de un programa.

- ✓ ***Condición simple****:* Es una variable lógica o una expresión relacional, representada de la siguiente forma. E1 < operador-relacional> E2 donde E1 y E2 son expresiones relacionales.
- ✓ ***Condición compuesta:*** Está formada por dos o más condiciones simples, operadores lógicos y paréntesis.

Prueba del flujo de datos

Es un método de diseño de pruebas con el objetivo de seleccionar los caminos de pruebas de un programa de acuerdo con las definiciones y los usos de las variables en él.

Prueba de Ciclos o Bucles

Esta prueba se centra solamente en verificar los ciclos presentes en el programa, existen cuatro tipos de ciclos que se definen como: simples, concatenados, anidados y no estructurados.

1.8.2 Método de prueba de caja negra

La Prueba de Naja Negra también denominada prueba de comportamiento se centra en los requisitos funcionales del software. Esta prueba permite al ingeniero de software obtener una serie de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Esta prueba intenta descubrir errores diferentes a los de la caja blanca. (Brito, 2003)

Los errores que son encontrados luego de ejecutar una prueba de caja negra se agrupan en las siguientes categorías:

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

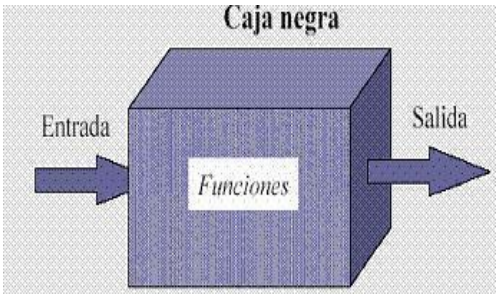

Figura 4: Método de Prueba de Caja Negra.

La Prueba de Caja Negra se aplica durante las fases posteriores a la prueba ya que esta ignora la estructura de control, basa más su atención en el campo de la información. Existen diferentes métodos para realizar una prueba de caja negra entre ellos:

Método de prueba basado en grafos.

El primer paso en la prueba de caja negra es entender los objetos que se modelan en el software y las relaciones que conectan a estos objetos, una vez que se haya realizado esto, el próximo paso es crear una serie de pruebas que verifiquen que los objetos tienen entre ellos, las relaciones esperadas. (Anna. C Grimán)

En fin la prueba del software lo primero que hace es crear un grafo de objetos y sus relaciones, luego se diseñan un grupo de pruebas que cubran todo el grafo con el objetivo de que se ejerciten todos los objetos y sus relaciones para detectar los errores.

Partición equivalente

Este es uno los métodos de caja negra más efectivos, divide el campo de entrada en un programa en clases de datos, mediante los cuales se derivan casos de pruebas, que permiten detectar errores inmediatamente. La partición equivalente permite además examinar todos los datos, tanto correctos como incorrectos, de las entradas existentes en un programa y descubrir errores. Este método reduce la cantidad de casos de pruebas que hay que realizar debido a que se dirige fundamentalmente a los casos de prueba que descubran casos de errores.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Si un conjunto de objetos puede unirse por medio de relaciones simétricas, transitivas y reflexivas, entonces existe una clase de equivalencia. Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Una condición de entrada es un valor numérico específico, un rango de valores relacionados o una condición lógica. (Anna. C Grimán)

El diseño de Casos de Prueba tiene como objetivo detectar los errores existentes en la aplicación sin consumir gran cantidad de recursos. Estas técnicas se encuentran mediatizadas por la imposibilidad de probar todas las posibilidades de funcionamiento del software. Por tal motivo la idea fundamental es

aplicar los casos de prueba a los Casos de Uso más críticos. De esta forma, se asume que si no se encuentran fallas al ejecutar los casos de prueba el producto se encuentra libre de defectos.

Análisis de valores límite

El análisis de valores límite es un diseño de casos de prueba que complementa la técnica de la partición equivalente. Conduce a una elección de los casos de prueba que ejerciten los valores límite. El objetivo de estas pruebas es comprobar que el sistema tiene la habilidad de manejar los datos que se encuentran en los límites aceptables.

1.9 Conclusiones

En este capítulo se sustentó teóricamente la investigación llevada a cabo. Fueron explicados los conceptos más importantes relacionados con el problema en cuestión. Se trataron los diferentes métodos y tipos de pruebas, así como sus niveles y la metodología de desarrollo a utilizar para lograr la realización de un buen procedimiento de pruebas. El cual es de gran importancia a la hora de construir un sistema de software, puesto que define una serie de actividades, que garantizan que el producto final sea entregado libre de defectos. Una vez realizada la investigación y definido los métodos, niveles y tipos de pruebas, se está en condiciones de realizar el diseño y aplicación de las mismas.

Capítulo 2: Diseño y Aplicación de las Pruebas

2.1 Introducción

En el presente capítulo se realiza el diseño y la aplicación de las pruebas al módulo Inventario de Aguas Minerales; con lo que se pretende detectar los errores presentes en el software. En esencia se precisa un Plan de Pruebas como base para todo el proceso de pruebas que se llevará a cabo, así como la estrategia de pruebas a seguir.

2.2 Descripción del Sistema a Probar

En Cuba, actualmente existen centros que enfocan su trabajo a proteger los recursos naturales. Específicamente se encuentra la Oficina Nacional de Recursos Naturales (ONRN), autoridad minero petrolera de la República de Cuba. Es la encargada de administrar el conocimiento y la información geológica, minera y petrolera de la nación. Debido a las necesidades de dicho centro se decidió llevar a cabo la realización de una aplicación WEB que permita realizar el proceso de forma automatizada. Dentro de las aplicaciones que conforman el sistema se encuentra el módulo Inventario de Aguas Minerales, el cual cuenta con 11 casos de uso y 14 requerimientos funcionales.

2.3 Características a probar

Las características a probar pueden ser variables en dependencia del software que se esté probando y lo que se pretenda evaluar con dicha prueba. Algunas de las características más importantes que se pretenden probar son: que el sistema implementado cumple con los requerimientos establecidos por el cliente. Además, se debe garantizar que el producto se encuentra libre de errores y que la documentación esté bien detallada, que la misma se corresponda con la aplicación desarrollada, y permita que el cliente pueda comprender y usar fácilmente el software.

Otras características más generales que se pueden tener en cuenta son relacionadas con el código de la aplicación por ejemplo si existen pruebas que permiten conocer si dicho código está bien estructurado y es además reutilizable. Se debe garantizar además que la aplicación tenga un buen funcionamiento, gran fiabilidad, usabilidad y seguridad, logrando con la verificación de estos aspectos que el sistema funcione de una forma correcta. En fin deben ser probados todos los aspectos del sistema, para lograr de esta forma que el producto llegue a la etapa final libre de defectos que puedan acortar su vida y ocasionar decepciones en los clientes.

2.4 Plan de Pruebas

El propósito del Plan de Pruebas es explicitar el alcance, enfoque, recursos requeridos, calendario, responsables y manejo de riesgos de un proceso de pruebas. [16]. Debe ser documentado y aprobado por el personal que interviene en la realización del mismo. En el Plan de Pruebas se especifican los recursos usados para hacer las pruebas, así como las estrategias que se llevarán a cabo en el diseño de los casos de prueba. En este se describen además los requisitos y los casos de uso, el personal responsable y los riesgos asociados.

El Plan de Pruebas es una guía para ejecutar las pruebas debido a que este es el primer paso que se debe tener en cuenta antes de realizarlas.

El plan de pruebas incluye:

Identificador del plan:

Como todo artefacto del desarrollo, está sujeto a control de configuración, por lo que debe distinguirse adicionalmente la versión y fecha del plan.

Alcance:

Indica el tipo de prueba y las propiedades o elementos del software a ser probados.

Estrategia:

Describe la técnica, patrón y/o herramientas a utilizarse en el diseño de los casos de prueba. En lo posible la estrategia debe precisar el número mínimo de casos de prueba a diseñar. La estrategia también muestra el grado de automatización que se exigirá, tanto para la generación de casos de prueba como para su ejecución.

Categorización de la configuración:

Muestra las condiciones bajo las cuales, el plan debe ser: suspendido, repetido o culminado. A veces el proceso de prueba debe suspenderse debido a errores que se han detectado. Luego de solucionar estas fallas, el proceso de prueba previsto por el plan puede continuar. Los criterios de culminación pueden ser tan simples como aprobar el número mínimo de casos de prueba diseñados o tan complejos como tomar en cuenta no sólo el número mínimo, sino también el tiempo previsto para las pruebas y la tasa de detección de fallas.

Recursos:

Especifica las propiedades necesarias y deseables del ambiente de prueba, incluyendo las características del hardware, el software o cualquier otro software necesario para llevar a cabo las pruebas. La sección incluye un estimado de los recursos humanos necesarios para el proceso.

Responsables:

Especifica quién es el responsable de cada una de las tareas previstas en el plan.

2.4.1 Objetivos

El objetivo general del plan de pruebas es establecer la cronología y condiciones para la aplicación de las pruebas de manera que permita obtener, un sistema que pueda ser completado con una recepción total de los interesados y entrar en operación con la totalidad de las funcionalidades requeridas para su funcionamiento. (2005)

DISEÑO Y APLICACIÓN DE LAS PRUEBAS

El objetivo que se persigue tras aplicar las pruebas al módulo Inventario de Aguas Minerales es ejecutar el programa atendiendo a un conjunto de casos de prueba que serán definidos posteriormente, con la intención de detectar y corregir errores que pueden estar presentes en el software, logrando mayor excelencia en el producto.

2.5 Estrategias de pruebas

Luego de haber estudiado minuciosamente las características del módulo Inventario de Aguas Minerales, se determinó que la prueba a nivel de Sistema es la más adecuada para el proceso de pruebas que se realizará. Esto se decidió principalmente por las características del producto a liberar, el cual brindará una interfaz visual, a través de la cual interactuará el usuario una vez integrado el sistema completamente. Las pruebas serán dirigidas a verificar que se han integrado adecuadamente todos los elementos del módulo y que el mismo realiza todas las funcionalidades requeridas.

Para lograr la efectividad de la estrategia de prueba a la hora de evaluar dinámicamente el software se debe comenzar primeramente por los componentes más simples y pequeños e ir avanzando hasta probar todo el sistema. Para esto se definen los pasos siguientes:

1. Pruebas unitarias.
2. Pruebas de Integración.
3. Pruebas de Sistema.
4. Pruebas de Aceptación.

2.6 Pasos a seguir en las pruebas

Tomando como punto de partida las características del módulo Inventario de Aguas Minerales se definieron las pruebas a realizar en los siguientes escenarios:

1. Aplicar Pruebas de Caja Blanca, específicamente la técnica del Camino Básico.

   La prueba se centra en encontrar los errores existentes en el código de la aplicación.

2. Pruebas de Caja Negra, específicamente la técnica de Partición Equivalente.

   Para esto se realiza la prueba de integración para comprobar que todas las funcionalidades funcionan correctamente como una unidad. Garantizando que todas las entradas tengan la salida esperada.

3. Aplicar pruebas de Validación.

   Esta prueba es aplicada para comprobar los criterios de validación, mediante esta prueba se puede asegurar que el producto satisface los requisitos funcionales, y de rendimiento.

4. Pruebas de Sistema.

   Por último, se realizarán pruebas al sistema para verificar la funcionalidad y el rendimiento total del producto.

2.7 Diseño de Casos de Prueba de Caja Negra

Casos de Prueba de Caja Negra para el módulo Inventario de Aguas Minerales

Nombre del Caso de Uso: Gestionar Yacimiento
Descripción General: El CU comienza cuando un Administrador desea adicionar, modificar o eliminar algún yacimiento. El sistema brinda las opciones de adición, búsqueda, modificación y eliminación al Administrador. Este selecciona la opción deseada, realiza la operación y termina el CU con la actualización de los datos.
Condiciones de Ejecución: El usuario tiene que estar autenticado con privilegios de administración para realizar cualquiera de las acciones antes mencionadas.

Tabla 1: Diseño de Caso de Prueba de Gestionar Yacimiento.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad

SC 1: Seleccionar Opción.	**EC 1.1:** Mostrar pestañas.	Se muestran dos pestañas “Buscar” y “Adicionar” para la adición, modificación, eliminación y búsqueda de yacimientos.
SC 2: Adicionar Yacimiento.	**EC 2.1**: Selección de la opción “Adicionar”.	El sistema muestra el formulario con los campos a llenar correspondiente a esta opción: Provincia, Municipio, Nombre, Fecha de Estimación. Así como un botón para ir a las páginas siguientes y adicionar las fuentes de agua que poseerá, así como los recursos disponibles de esta.
	EC 2.2: Introducir datos del yacimiento	El sistema crea un nuevo yacimiento y lo muestra en la lista de los yacimientos existentes que contiene la página de la opción “Buscar”.
	EC 2.3: Dejar campos en blanco o introducir datos	El sistema marca en rojo los campos que no han sido introducidos o que presentan datos incorrectos.
	EC: 2.4: Existencia del yacimiento.	El sistema muestra un mensaje informando al usuario sobre la existencia de ese yacimiento “Este yacimiento ya ha sido adicionado anteriormente”.
	EC: 2.5: Cancelar adición.	El sistema muestra un mensaje de confirmación “¿Está seguro que desea salir?”. De manera que si es aceptado el mensaje, se muestra la pestaña “Buscar” con todo el listado de los yacimientos existentes.
SC 3: Buscar Yacimiento.	**EC 3.1:** Listar todos los yacimientos.	Se muestra un listado con todos los yacimientos.

	EC 3.2: Búsqueda con resultados.	Se permite la entrada de un valor de búsqueda y se deben mostrar los resultados obtenidos de acuerdo a dicho valor.
	EC 3.3: Búsqueda sin resultados.	Se permite la entrada de un valor de búsqueda, pero al no obtenerse resultados para este valor se debe mostrar un mensaje que informe de ello al usuario.
	EC 3.4: No se introduce texto alguno.	Se muestran todos los yacimientos existentes hasta ese momento.
	EC 3.5: Generar documento Word.	Se genera un documento Word con los resultados obtenidos en una búsqueda determinada.
	EC 3.6: Generar documento PDF.	Se genera un documento PDF con los resultados obtenidos en una búsqueda determinada.
	EC 3.7: Utilizar paginado.	Se debe poder avanzar o retroceder en la visualización de los resultados obtenidos.
SC 4: Modificar Yacimiento.	**EC 4.1:** Selección de la opción “Modificar”.	El sistema muestra el panel con un formulario que contiene los mismos datos descritos en el **EC 2.1** del Yacimiento para que el usuario modifique el/los que desea.
	EC 4.2: El usuario modifica datos correctamente.	El sistema modifica los datos del yacimiento y muestra el listado de yacimientos.
	EC 4.3 El usuario modifica datos en los campos pero incorrectamente.	El sistema marca en rojo los campos que presentan datos incorrectos.

	EC 4.4: Cancelar la acción.	El sistema mantiene los datos del Yacimiento y vuelve a la pestaña “Buscar”.
SC 5: Eliminar Yacimiento.	**EC 5.1:** Eliminar datos.	El sistema muestra un mensaje de confirmación “¿Está seguro que desea eliminarlo?”
	EC 5.2: Aceptar.	El sistema elimina el yacimiento y actualiza la lista de estos.
	EC 5.3: Cancelar.	El sistema no elimina el yacimiento.

Nombre del Caso de Uso: Gestionar Fuente de Agua

Descripción General: El CU comienza cuando el Administrador desea adicionar, buscar, modificar o eliminar una fuente de agua, ya sea un pozo o un manantial. El sistema brinda las opciones pertinentes. El Administrador selecciona la opción deseada, realiza la operación y termina el CU con la actualización de los datos de dicha fuente de agua.

Condiciones de Ejecución: El usuario tiene que estar autenticado con privilegios de administración para realizar cualquiera de las acciones antes mencionadas.

Tabla 2: Diseño de Caso de Prueba de Gestionar Fuente de Agua.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Seleccionar Opción.	**EC 1.1:** Mostrar pestañas.	Se muestran dos pestañas “Buscar” y “Adicionar” para la adición, modificación, eliminación y búsqueda de una fuente de agua.

SC 2: Adicionar Fuente de Agua.	**EC 2.1**: Selección de la opción "Adicionar".	El sistema muestra el formulario con los campos a llenar correspondiente a esta opción: Yacimiento, Denominación, Abatimiento máximo, tipo de fuente, Sectores y Concesión. Así como un botón para ir a la página siguiente y adicionar los
	EC 2.2: Introducir datos de la fuente de agua correctamente.	El sistema crea una nueva fuente de agua y la muestra en la lista de las fuentes de agua existentes que contiene la pestaña "Buscar".
	EC 2.3: Dejar campos en blanco o introducir datos de la fuente de agua de forma	El sistema marca en rojo los campos que no han sido introducidos o que presentan datos incorrectos.
	EC: 2.4: Existencia de la fuente de agua.	El sistema muestra un mensaje informando al usuario sobre la existencia de esa fuente de agua "Esta fuente de Agua ya ha sido
	EC: 2.5: Cancelar adición.	El sistema muestra un mensaje de confirmación "¿Está seguro que desea salir?". De manera que si es aceptado el mensaje, se muestra la pestaña "Buscar" con todo el listado de las fuentes de agua existentes.
SC 3: Buscar Fuente de Agua.	**EC 3.1:** Listar todas las fuentes de agua.	Se muestra un listado con todas las fuentes de agua.
	EC 3.2: Búsqueda con resultados.	Se permite la entrada de un valor de búsqueda y se deben mostrar los resultados obtenidos de acuerdo a dicho valor.

	EC 3.3: Búsqueda sin resultados.	Se permite la entrada de un valor de búsqueda, pero al no obtenerse resultados para este valor se debe mostrar un mensaje que informe de ello al usuario.
	EC 3.4: No se introduce texto alguno.	Se muestran todas las fuentes de agua existentes hasta ese momento.
	EC 3.5: Generar documento Word.	Se genera un documento Word con los resultados obtenidos en una búsqueda determinada.
	EC 3.6: Generar documento PDF.	Se genera un documento PDF con los resultados obtenidos en una búsqueda determinada.
	EC 3.7: Cantidad de resultados.	Se debe visualizar en cada pantalla la cantidad de resultados obtenidos, indicada por el usuario.
	EC 3.8: Utilizar paginado.	Se debe poder avanzar o retroceder en la visualización de los resultados obtenidos.
SC 4: Modificar Fuente de Agua.	**EC 4.1:** Selección de la opción “Modificar”.	El sistema muestra el panel con un formulario que contiene los mismos datos descritos en el **EC 2.1** de la Fuente de Agua para que el usuario modifique el/los que desea.

	EC 4.2: El usuario modifica datos correctamente.	El sistema actualiza los datos de la fuente de agua y muestra el listado con todas las fuentes de agua existentes.
	EC 4.3: El usuario modifica datos en los campos pero incorrectamente.	El sistema marca en rojo los campos que presentan datos incorrectos.
	EC 4.4: Cancelar la acción.	El sistema mantiene los datos de la fuente de agua y vuelve a la pestaña “Buscar”.
SC 5: Eliminar Fuente de Agua.	**EC 5.1:** Eliminar datos.	El sistema muestra un mensaje de confirmación “¿Está seguro que desea eliminarla?”
	EC 5.2: Aceptar.	El sistema elimina la fuente de agua y actualiza la lista de estas.
	EC 5.3: Cancelar.	El sistema no elimina la fuente de agua.

Nombre del Caso de Uso: Gestionar Recurso Disponible

Descripción General: El CU comienza cuando el Administrador desea adicionar, buscar, modificar o eliminar un recurso disponible. El sistema brinda las opciones pertinente. El Administrador selecciona la opción deseada, realiza la operación y termina el CU con la actualización de los datos de dicho recurso.

Condiciones de Ejecución: El usuario tiene que estar autenticado con privilegios de administración para realizar cualquiera de las acciones antes mencionadas.

Tabla 3: Diseño de Caso de Prueba de Gestionar Recurso Disponible.

Nombre de la	Escenarios de la sección	Descripción de la funcionalidad

sección		
SC 1: Seleccionar Opción.	EC 1.1: Mostrar pestañas.	Se muestran dos pestañas “Buscar” y “Adicionar” para la adición, modificación, eliminación y búsqueda de un recurso disponible.
SC 2: Adicionar Recurso Disponible.	EC 2.1: Selección de la opción “Adicionar”.	El sistema muestra el formulario con los campos a llenar correspondiente a esta opción: Yacimiento, Fuente, Materia Prima, Mineralización, PH, Temperatura, Fecha, Medidos, Indicados, Inferidos, Sectores, Macroelementos (HCO3, SO4, Cl, Ca, Na, K, Mg), Microelemento (H2O, H2SiO3, NO3, NO2),
	EC 2.2: Introducir datos del recurso disponible correctamente.	El sistema crea un nuevo recurso disponible y lo muestra en la lista de los recursos disponibles existentes que contiene la pestaña “Buscar”.
	EC 2.3: Dejar campos en blanco o introducir datos del recurso disponible de forma	El sistema marca en rojo los campos que no han sido introducidos o que presentan datos incorrectos.
	EC: 2.4: Cancelar adición.	El sistema muestra un mensaje de confirmación “¿Está seguro que desea salir?”. De manera que si es aceptado el mensaje, se muestra la pestaña “Buscar” con todo el listado de las fuentes de agua existentes.
SC 3: Buscar Recurso Disponible.	EC 3.1: Listar todas los recursos disponibles.	Se muestra un listado con todos los recursos disponibles.

	EC 3.2: Búsqueda con resultados.	Se permite la entrada de un valor de búsqueda y se deben mostrar los resultados obtenidos de acuerdo a dicho valor.
	EC 3.3: Búsqueda sin resultados.	Se permite la entrada de un valor de búsqueda, pero al no obtenerse resultados para este valor se debe mostrar un mensaje que informe de ello al usuario.
	EC 3.4: No se introduce texto alguno.	Se muestran todos los recursos disponibles existentes hasta ese momento.
	EC 3.5: Generar documento Word.	Se genera un documento Word con los resultados obtenidos en una búsqueda determinada.
	EC 3.6: Generar documento PDF.	Se genera un documento PDF con los resultados obtenidos en una búsqueda determinada.
	EC 3.7: Cantidad de resultados.	Se debe visualizar en cada pantalla la cantidad de resultados obtenidos, indicada por el usuario.
	EC 3.8: Utilizar paginado.	Se debe poder avanzar o retroceder en la visualización de los resultados obtenidos.
SC 4: Modificar Recurso Disponible.	EC 4.1: Selección de la opción “Modificar”.	El sistema muestra el panel con un formulario que contiene los mismos datos descritos en el EC 2.1 del Recurso Disponible para que el usuario modifique el/los que desea.
	EC 4.2: El usuario modifica datos correctamente.	El sistema muestra un mensaje de confirmación “¿Realmente desea modificar este Recurso Disponible?”.

	EC 4.3: Aceptar acción.	El sistema actualiza los datos de la fuente de agua y actualiza el listado en la pestaña “Buscar”.
	EC 4.4 El usuario modifica datos en los campos pero incorrectamente.	El sistema marca en rojo los campos que presentan datos incorrectos.
	EC 4.5: Cancelar la acción.	El sistema mantiene los datos de la fuente de agua y vuelve a la pestaña “Buscar”.
SC 5: Eliminar Recurso Disponible.	EC 5.1: Eliminar datos.	El sistema muestra un mensaje de confirmación “¿Está seguro que desea eliminarla?”
	EC 5.2: Aceptar.	El sistema elimina el recurso disponible y actualiza la lista de estos.
	EC 5.3: Cancelar.	El sistema no elimina el recurso disponible.
SC 6: Ver detalles de Recurso Disponible.	EC 6.1: Selección de la opción “Ver detalles”.	El sistema muestra una ventana de diálogo donde visualiza todos los campos que fueron descritos en EC 2.1 y que fueron entrados por el usuario para el Recurso Disponible seleccionado.

2.8 Diseño de Pruebas de Caja Blanca. Técnica: Camino Básico.

Caso de Prueba de Caja Blanca para el módulo Inventario de Aguas Minerales

Caso de Prueba del Caso de Uso Gestionar Yacimiento.

Paso1. Dibujar el Grafo de Flujo o Grafo del Programa a partir del código fuente.

```
public function executeBuscarYacimiento() {

1 $texto = $this->getRequestParameter('txtTexto');

1 $inicio = $this->getRequestParameter('txtInicio');

1 $cantidad = $this->getRequestParameter('txtCantidad');

1 $yacimientos = TyacimientoaguaPeer::BuscarYaciminetoTexto($texto, $cantidad);

1 $total = TyacimientoaguaPeer::CantidadYaciminetoTexto($texto);

1 $Json = array();

1 $i = 0;

2 foreach ($yacimientos as $elem){

3 $Json[$i++] = array(

4 "Nombre" => $elem["nombre"],

4 "Provincia" => $elem["pnombre"],

4 "Municipio" => $elem["mnombre"],

4 "Fecha" => $elem["fecha"],

4 "id"=> $elem["id"] ); }

5 $titulos = array( 0 =>"Nombre", 1 => "Provincia", 2 => "Municipio", 3 => Fecha",

4 => "Acciones", 5 => "1", 6 => "2", 7 => "3" );
```

```
5 $acciones= array(
6 Array(
6 "img" => '../../../../images/icon/edit.png'","accion" => “/balance/yacimiento/Actualizar Yacimiento/id/",)
6 Array(
6 "img" => '<img src="../../../../images/icon/delete.png'",
6 "confirmar" => "1",
6 "accion" => "/balancea/yacimiento/Eliminar/id/",
6 "mensaje" => "Estas seguro que desea eliminarlo",
6 "titulo" => "Eliminar Yacimiento”));
6 $Json[$i++]= $acciones;
6 $Json[$i++] = $titulos;
6 $Json[$i++] = array("total" => $total);
6 $array_json = json_encode($Json);
7 $this->getResponse()->setHttpHeader("application/json");
8 echo $array_json;
9        return sfView::NONE; }
```

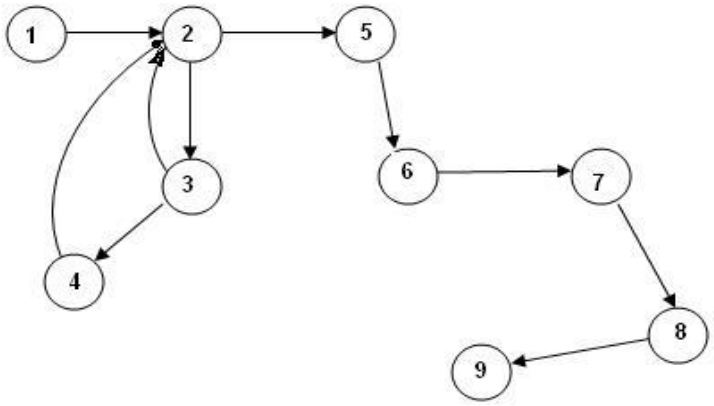

Figura 5: Grafo de Flujo del Método Buscar Yacimiento

Paso 2. Calcular la Complejidad Ciclomática.

V (G) = Número de Aristas – Número de Nodos + 2

V (G) = 10 – 9 + 2

V (G) = 3

Paso 3. Determinar los Caminos Independientes.

Caminos Básicos

CB1: 1 – 2 – 3 – 4 – 2 – 5 – 6 – 7 – 8 – 9

CB2: 1 – 2 – 5 – 6 – 7 – 8 – 9

CB3: 1 – 2 – 3 – 2 – 5 – 6 – 7 – 8 – 9

Paso 4 Casos de Prueba del Camino Básico.

CB1: 1 – 2 – 3 – 4 – 2 – 5 – 6 – 7 – 8 – 9

Caso de Prueba: Buscar yacimiento

Entrada: Buscar = Mineral.

Resultado Esperado: Se encuentra el yacimiento buscado

Resultado de la prueba: Satisfactorio.

Caso de Prueba del Caso de Uso Gestionar Fuente de Agua

Paso1. Dibujar el Grafo de Flujo o Grafo del Programa a partir del código fuente.

```
public function executeBuscarFuenteAgua() {
 1 $texto = $this->getRequestParameter('txtTexto');
 1 $inicio = $this->getRequestParameter('txtInicio');
 1 $cantidad = $this->getRequestParameter('txtCantidad');
 1 $fuentes = TfuenteaguaPeer::BuscarFuenteAguaTexto($texto, $cantidad, $inicio);
 1 $total = TfuenteaguaPeer::CantidadFuenteAguaTexto($texto);
 1 $Json = array();
 1        $i = 0;
 2        foreach ($fuentes as $elem){
 3 $Json[$i++] = array (
```

```
4         "Yacimiento" => $elem["nombreyac"],
4         "Nombre" => $elem["nombrefuente"],
4         "Abatimiento Max" => $elem["abat"],
4         "id" => $elem["id"] );}
5 $titulos = array(
6                 0 =>"Yacimiento",
6        1 => "Nombre",
6        2 => "Abatimiento Max",
6        3 => "Acciones",
6        4 => "1",
6        5 => "2");
7 $acciones= array(
8         array(
8                 "img" => '../../../../images/icon/edit.png"',
8        "accion" => "/balancea.php/fuenteagua/Actualizar/id/"),
8         array(
8        "img" => '<img src="../../../../images/icon/delete.png"',
```

8 "confirmar" => "1",

8 "accion" => "/balancea.php/fuenteagua/Eliminar/id/",

8 "mensaje" => "Estas seguro que desea eliminarlo",

8 "titulo" => "Eliminar Prospecto"));

9 $Json[$i++]= $acciones;

9 $Json[$i++] = $titulos;

9 $Json[$i++] = array("total" => $total);

10 $this->getResponse()->setHttpHeader("application/json");

11 $array_json = json_encode($Json);

12 echo $array_json;

13 return sfView::NONE; }

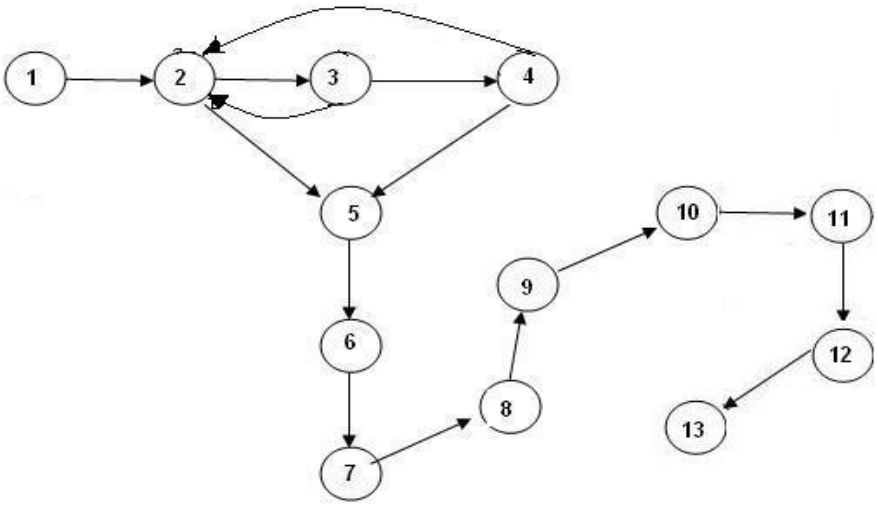

Figura 6: Grafo de Flujo del Método Buscar Fuente de Agua.

Paso 2. Calcular la Complejidad Ciclomática.

V (G) = Número de Aristas – Número de Nodos + 2

V (G) = 15 – 13 + 2

V (G) = 4

Paso 3. Determinar los Caminos Independientes.

Caminos Básicos

CB1: 1 – 2 – 3 – 4 – 2 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13

CB2: 1 – 2 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13

CB3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13

Paso 4 Casos de Prueba del Camino Básico.

CB1: 1 – 2 – 3 – 4 – 2 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13

Caso de Prueba: Buscar Fuente de Agua

Entrada: Buscar = Y

Resultado Esperado: Se busca una fuente de agua que contiene un yacimiento llamado mineral

Resultado de la Prueba: Satisfactorio.

Caso de Prueba del Caso de Uso Gestionar Recurso Disponible

Paso1. Dibujar el Grafo de Flujo o Grafo del Programa a partir del código fuente.

```
public function executeEliminar() {
 1      if($this->getRequest()->getMethod() != sfRequest::POST) {
 2       return sfView::SUCCESS; }
else {
 3       $id = $this->getRequestParameter('id');
 3       TfuenteaguaPeer:Eliminar($id);
 3       $this->redirect('fuenteagua/fuenteAgua'); }
 4 }
```

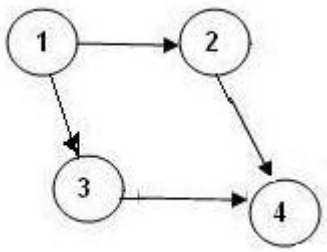

Figura 7: Grafo de Flujo del Método Eliminar Recurso Disponible

Paso 2. Calcular la Complejidad Ciclomática.

V (G) = Número de Aristas – Número de Nodos + 2

V (G) = 4 – 4 + 2

V (G) = 2

Paso 3. Determinar los Caminos Independientes.

Caminos Básicos

CB1: 1 – 2 – 4

CB2: 1 – 3 – 4

Paso 4 Casos de Prueba del Camino Básico.

CB1: 1 – 2 – 4

Caso de Prueba: Eliminar Recurso Disponible

Entrada: Yacimiento = X, Fuente = A, Caudal Medio = 1, Caudal Indicado = 1, Caudal Inferido = 1

Resultado Esperado: Se elimina el recurso disponible seleccionado

Resultado de la Prueba: No Satisfactorio

Comentario: El recurso es eliminado pero no se actualiza el listado.

2.9 Conclusiones

En este capítulo se diseñaron los casos de pruebas pertinentes al módulo Inventario de Aguas Minerales y fueron realizadas específicamente las pruebas de Caja Blanca. Quedó elaborado el plan de prueba correspondiente a este módulo y fue definida la estrategia que guiará el proceso de pruebas para comprobar el buen funcionamiento del sistema. Al aplicar las pruebas se realizó una revisión completa al sistema implementado, mediante lo cual se detectaron los errores existentes en el código.

Capítulo 3: Análisis de los Resultados Obtenidos

3.1 Introducción

Una vez aplicadas las pruebas al módulo Inventario de Aguas Minerales es necesario documentar el resultado de las mismas, puesto que puede servir como ayuda para mejorar la calidad de futuros productos de software. En el presente capítulo se lleva a cabo un análisis de los resultados obtenidos luego de aplicar los Casos de Pruebas que fueron diseñados para el módulo Inventario de Aguas Minerales.

3.2 Resultados de las pruebas a la documentación del módulo Inventario de Aguas Minerales

A la hora de realizar las pruebas uno de los aspectos fundamentales que debe ser revisado es la documentación del software. El hecho de que la documentación esté correcta da la posibilidad de tener un control estricto de todos los casos de prueba que se van a aplicar, ya sea para las pruebas de Caja Blanca o las pruebas de Caja Negra. Conjuntamente con la liberación del módulo serán liberados también los documentos:

1. Especificación de Requisitos.
2. Manual de Usuario.

Existieron algunas dificultades a la hora de realizar la revisión de los mismos puesto que el manual de usuario aún no está confeccionado y fueron detectadas una serie de no conformidades en la especificación de requisitos las cuales serán mencionadas a continuación:

- ✓ No han sido identificadas las restricciones de la implementación.
- ✓ Los números de página que aparecen en el índice no coinciden con el contenido que se refleja realmente en las páginas.

- ✓ El número de páginas que aparece definido en las reglas de confidencialidad no coincide con el total de páginas que realmente tiene el documento.
- ✓ El sistema no proporciona un soporte adecuado para usuarios principiantes.

Para la revisión de la documentación existe una técnica muy efectiva conocida como Listas de Chequeo o Listas de Defectos. Estas son bien conocidas y utilizadas debido a que permiten tener una idea previa de los posibles errores que se pueden encontrar en el software. Por tanto, son muy beneficiosas para el personal encargado de realizar las pruebas.

3.3 Resultados obtenidos de aplicar las pruebas de Caja Blanca y Caja Negra al módulo Inventario de Aguas Minerales

3.3.1 Funcionalidad: Gestionar Yacimiento

Tabla 4: Resultados de Gestionar Yacimiento.

Secciones	No	No conformidad	Etapa de detección	Signif	No Signif	Recomendación
Adicionar Yacimiento	1	El campo nombre acepta la entrada de números como dato válido.	Prueba	X		Validar correctamente los datos de entrada del campo nombre
	2	Los botones enviar y cancelar no tienen nombre, pero funcionan correctamente.	Prueba		X	Ponerle nombre a los botones enviar y cancelar para un lograr un mejor entendimiento del usuario

Modificar Yacimiento	3	El sistema no modifica el yacimiento da un error al intentar modificar.	Prueba	X		Esta funcionalidad debe ser implementada correctamente.
Eliminar Yacimiento	4	El sistema elimina el yacimiento pero no actualiza el listado luego de haberlo eliminado.	Prueba	X		La funcionalidad actualizar debe ser implementada correctamente.

3.3.2 Funcionalidad: Gestionar Fuente de Agua

Tabla 5: Resultado de Gestionar Fuente de Agua.

Secciones	No	No conformidad	Etapa de detección	Signif	No Signif	Recomendación
Adicionar fuente de agua	1	El campo Concesión está mal escrito	prueba		X	Las faltas de ortografía atentan contra la estética del producto por tanto es necesario revisar este error y corregirlo.
	2	Los botones enviar y	prueba		X	Los botones enviar y cancelar deben tener un

		cancelar no tienen nombre				nombre que los identifique
Modificar Fuente de Agua	3	El sistema no modifica la fuente de agua da un error al intentar modificar.	prueba	X		Esta funcionalidad debe ser implementada correctamente.
Eliminar Fuente de Agua	4	El sistema elimina la fuente de agua pero no actualiza el listado luego de haberla eliminado.	prueba	X		La funcionalidad actualizar debe ser implementada correctamente.

3.3.3 Funcionalidad: Gestionar Recurso Disponible

Tabla 6: Resultado de Gestionar Recurso Disponible.

Secciones	No	No conformidad	Etapa de detección	Signif	No Signif	Recomendación

Modificar Recurso Disponible	1	El sistema no modifica el recurso disponible da un error al intentar modificar.	prueba	X		Esta funcionalidad debe ser implementada correctamente.
Eliminar Recurso Disponible	2	El sistema elimina el recurso disponible pero no actualiza el listado luego de haberlo eliminado.	prueba	X		La funcionalidad actualizar debe ser implementada correctamente.
Ver detalles del Recurso Disponible	3	No se muestra el cuadro de diálogo, por tanto el usuario no puede ver los detalles del recurso disponible.	prueba	X		Esta funcionalidad debe ser implementada correctamente.

3.3.4 Funcionalidad: Gestionar Recurso de Explotación

Tabla 7: Resultado de Gestionar Recurso de Explotación.

Secciones	No	No conformidad	Etapa de detección	Signif	No Signif	Recomendación
Modificar Recurso de Explotación	1	El sistema no modifica el recurso de explotación, da un error al intentar modificar.	prueba	X		Esta funcionalidad debe ser implementada correctamente.
Eliminar Recurso de Explotación	2	El sistema elimina el recurso de explotación pero no actualiza el listado luego de haberlo eliminado.	prueba	X		La funcionalidad actualizar debe ser implementada correctamente.

3.3.5 Funcionalidad: Gestionar Recurso en Explotación

Tabla 8: Resultado de Gestionar Recurso en Explotación**.**

Secciones	No	No conformidad	Etapa de detección	Signif	No Signif	Recomendación

Modificar Recurso en Explotación	1	El sistema no modifica el recurso en explotación, da un error al intentar modificar.	prueba	X		Esta funcionalidad debe ser implementada correctamente.
Eliminar Recurso en Explotación	2	El sistema elimina el recurso en explotación pero no actualiza el listado luego de haberlo eliminado.	prueba	X		La funcionalidad actualizar debe ser implementada correctamente.

3.3.6 Funcionalidad: Gestionar Control Externo.

Tabla 9: Resultado de Gestionar Control Externo.

Secciones	No	No conformidad	Etapa de detección	Signif	No Signif	Recomendación

Modificar Control Externo	1	El sistema no modifica el control externo, da un error al intentar modificar.	prueba	X		Esta funcionalidad debe ser implementada correctamente.
Eliminar Control Externo	2	El sistema elimina el control externo pero no actualiza el listado luego de haberlo eliminado.	prueba	X		La funcionalidad actualizar debe ser implementada correctamente.

Luego de analizar cada uno de los resultados obtenidos en la ejecución de las pruebas de Caja Blanca y Caja Negra se puede resumir que algunos de los errores presentes en el software no son muy significativos a la hora de evaluar el producto, pues son muy fáciles de corregir y no afectan el funcionamiento de la aplicación, como las faltas de ortografía que no conducen a fallas en el sistema pero si atentan contra la estética y la calidad del software. No ocurre igual con otros de los defectos que fueron detectados y se determinó que tienen gran importancia ya que pueden conllevar a que el sistema en algún momento no funcione de acuerdo con los requerimientos del cliente. Por otro lado, el manual de usuario debe ser confeccionado puesto que es la guía que tiene el usuario para lograr un entendimiento con el software.

ÁLISIS DE LOS RESULTADOS OBTENIDOS

3.4 Evaluación del módulo Inventario de Aguas Minerales

La evaluación del módulo Inventario de Aguas Minerales se realizó teniendo en cuenta los siguientes atributos de calidad:

- ✓ Seguridad
- ✓ Usabilidad
- ✓ Portabilidad
- ✓ Confiabilidad

Para comprobar estos aspectos se tuvo en cuenta una lista de chequeo para cada uno por separado, esta lista cuenta con una serie de preguntas relacionadas con los atributos de calidad, a las que le fueron asignadas un peso según su importancia y un valor según el cumplimiento de la pregunta en el sistema. A partir de los resultados cuantitativos obtenidos una vez aplicada la lista de chequeo, el módulo Inventario de Aguas Minerales puede clasificarse como:

Seguro

Este sistema brinda seguridad a la información que almacena debido a que permite acceder a la aplicación sólo a personas previamente autorizadas y éstas a su vez pueden realizar sólo las acciones definidas para su rol.

Poco usable

Aunque es un sistema que no exige de avanzados conocimientos sobre sistemas informáticos para utilizarlo, se considera poco usable por no contar con la documentación suficiente como ayuda para los usuarios que interactúen con él, además no cuenta con un mecanismo de ayuda en la aplicación, este atributo de calidad debe tenerse en cuenta desde los primeros momentos en los que se empieza a realizar un software.

Portable

El sistema es multiplataforma, garantiza su correcto funcionamiento independientemente del software o hardware utilizado.

Confiabilidad

Este sistema se considera confiable debido a que responde correctamente ante las fallas que ocurran durante su ejecución, garantizando que el usuario pueda contar con los servicios que brinda la aplicación durante el tiempo que lo necesite.

3.5 Conclusiones

En este capítulo se llevó a cabo el análisis de los resultados obtenidos luego de haber aplicado las pruebas al módulo Inventario de Aguas Minerales. Se puede asegurar que el proceso fue realizado de forma satisfactoria, debido a que los casos de prueba ayudaron a detectar los errores existentes en la aplicación. La utilización de las listas de chequeo a la hora de revisar la documentación fue muy efectiva permitiendo que se arrojaran no conformidades que influyen directamente en la aceptación del producto final.

Conclusiones

El proceso de pruebas llevado a cabo sobre un software garantiza organización, seguridad e integridad del producto. El principal objetivo de las pruebas es velar por la calidad del software, por este motivo aplicar pruebas a los productos en la etapa de desarrollo es un factor de gran importancia para garantizar la calidad del producto final y la satisfacción del cliente. Se realizaron pruebas sobre la documentación perteneciente al módulo Inventario de Aguas Minerales, con lo que se pretende garantizar la entrega de una documentación libre de errores que le permita al usuario interactuar fácilmente con el software. Fueron diseñados once casos de pruebas, respondiendo a la cantidad de casos de usos del sistema, lo que permitió probar cada una de las funcionalidades y documentar los errores encontrados tanto en el código como en la interfaz. Quedó confeccionado un plan de pruebas, donde se especifican los recursos necesarios para hacer las pruebas así como la estrategia de prueba a seguir. Las pruebas aplicadas al módulo Inventario de Aguas Minerales fueron realizadas exitosamente partiendo del principio “El éxito de una prueba radica precisamente en detectar un error no visto hasta entonces”.

Recomendaciones

Una vez realizado el proceso de prueba sobre el módulo Inventario de Aguas Minerales perteneciente al proyecto Sistema de Gestión de Datos Geológicos se recomienda:

1. Para próximas versiones del proyecto se recomienda emplear la herramienta Selenium IDE para llevar a cabo el proceso de pruebas al software.

2. Hacer una revisión completa a la documentación del software para garantizar que esta se corresponda con la aplicación.

3. Confeccionar el manual de usuario.

Referencias Bibliográficas

Anna. C Grimán, María Pérez, Luis. E Mendoza. Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales. Caracas Venezuela : s.n.

Brito, Irina Reyes. 2003. Las pruebas de Software. 2003.

Collazo, Manuel. 1990. Técnicas de pruebas del software. Estrategias de prueba del software. . 1990. IEEE Standard Glossary of Software Engineering Terminology,.

Delgado, Erly. 2010. Metodologías de Desarrollo de Software. 2010.

2005. El plan de pruebas. [En línea] Enero de 2005. [Citado el: 15 de Enero de 2010.] http://www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html .

Fabian. 2007. Etapa de pruebas. [En línea] 16 de Septiembre de 2007. [Citado el: 15 de Abril de 2010.] http://fabian-quintosemestre.blogspot.com/2007/09/etapa-de-pruebas.html.

Fernández, Carlos Alberto. 2000. El proceso Unificado de Rational para el Desarrollo de Software. 2000.

Hugo, Ing.Roberto. Introduccción a la Calidad del Software. [En línea] [Citado el: 25 de Marzo de 2010.] http://grictics.frm.utn.edu.ar/docs/introducción%2a/%20Calidad%de20Software%20Vazquez.pdf.

2009. IEEE. ESTRATEGIA DE PRUEBAS. [En línea] 6 de Dicimebre de 2009. http://www.udistrital.edu.co/comunidad/grupos/arquisoft/index.php?id=81&type.

Ivar Jacobson, Grady Booch, James Rumbaugh. 2000. El Proceso Unificado de Desarrollo de Software. España : PEARSON EDUCACION, 2000. ISBN.

Lovelle, Juan Manuel Cueva. 1999. Calidad del software . Oviedo España : s.n., 1999.

Mañas, José A. Prueba de programas. [En línea] [Citado el: 15 de Enero de 2010.] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s7.

Pressman. 2005. Un enfoque práctico. 5ta Edición. 2005.

Pressman, R.S. (1998). Ingeniería del Software. Un enfoque práctico. 4a Edición. (1998).

2009. Pruebas de software. [En línea] 5 de Diciembre de 2009. http://lsi.ugr.es/~ig1/docis/pruso.pdf.

Soto, Lauro. Calidad de Software. [En línea] [Citado el: 25 de Marzo de 2010.]

Bibliografía

1. Anna. C Grimán, María Pérez, Luis. E Mendoza. Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales. Caracas Venezuela : s.n.
2. Brito, Irina Reyes. 2003. Las pruebas de Software. 2003.
3. Collazo, Manuel. 1990. Técnicas de pruebas del software. Estrategias de prueba del software. . 1990. IEEE Standard Glossary of Software Engineering Terminology.
4. Delgado, Erly. 2010. Metodologías de Desarrollo de Software. 2010.
5. Documento Arquitectura de Software. Proyecto Sistema de Gestión de Datos Geológicos. V1.1.
6. Documento Modelo de Caso de Uso del Sistema. Proyecto Sistema de Gestión de Datos Geólogicos. V1.1.
7. 2005. El plan de pruebas. [En línea] Enero de 2005. [Citado el: 15 de Enero de 2010.] http://www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html.
8. Fabian. 2007. Etapa de pruebas. [En línea] 16 de Septiembre de 2007. [Citado el: 15 de Abril de 2010.] http://fabian-quintosemestre.blogspot.com/2007/09/etapa-de-pruebas.html.
9. Fernández, Carlos Alberto. 2000. El proceso Unificado de Rational para el Desarrollo de Software. 2000.
10. Gelmer. Analisis y desarrollo de sistemas de informacion. Junio 5, 2009. http://desasof2004.blogspot.com/2009/06/plan-de-pruebas-de-software.html (accessed Febrero 3, 2010).
11. Hugo, Ing.Roberto. Introduccción a la Calidad del Software. [En línea] [Citado el: 25 de Marzo de 2010.] http://grictics.frm.utn.edu.ar/docs/introducción%2a/%20Calidad%de20Software%20Vazquez.pdf.
12. 2009. IEEE. ESTRATEGIA DE PRUEBAS. [En línea] 6 de Dicimebre de 2009. http://www.udistrital.edu.co/comunidad/grupos/arquisoft/index.php?id=81&type.

BIBLIOGRAFÍA

13. Isabel Blank, L. H. (2005). Pruebas de Funcionalidad.

14. Ivar Jacobson, Grady Booch, James Rumbaugh. 2000. El Proceso Unificado de Desarrollo de Software. España : PEARSON EDUCACION, 2000. ISBN.

15. J.lturrioz. 2005. Técnicas de Evaluación Dinámica

16. Lovelle, Juan Manuel Cueva. 1999. Calidad del software . Oviedo España : s.n., 1999.

17. Mañas, José A. Prueba de programas. [En línea] [Citado el: 15 de Enero de 2010.] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s7.

18. Pressman. 2005. Un enfoque práctico. 5ta Edición. 2005.

19. Pressman, R.S. (1998). Ingeniería del Software. Un enfoque práctico. 4a Edición. (1998).

20. 2009. Pruebas de software. [En línea] 5 de Diciembre de 2009. http://lsi.ugr.es/~ig1/docis/pruso.pdf.

21. Soto, Lauro. Calidad de Software. [En línea] [Citado el: 25 de Marzo de 2010.].

Anexos

Anexo 1 No conformidades detectadas en la documentación.

Elemento	No	No conformidad	Etapa de la detención
Manual de usuario	1	El documento no está confeccionado	Pruebas a la documentación
Especificación de requerimientos	2	No han sido identificadas las restricciones de la implementación.	Pruebas a la documentación
Especificación de requerimientos	3	Los números de página que aparecen en el índice no coinciden con el contenido que se refleja realmente en las páginas.	Pruebas a la documentación
Especificación de requerimientos	4	El número de páginas que aparece definido en las reglas de confidencialidad no coincide con el total de páginas que realmente tiene el documento.	Pruebas a la documentación
Especificación de requerimientos	5	El sistema no proporciona un soporte adecuado para usuarios principiantes.	Pruebas a la documentación

Anexo 2 Lista de chequeo aplicada a la documentación.

No	Nombre del documento.	Evaluación	Fecha
1	Especificación de requerimiento.	¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?	2/4/2010
2	Especificación de requerimiento.	Los números de página que aparecen en el índice coinciden con el contenido que se refleja en las páginas.	2/4/2010
3	Especificación de requerimiento.	¿Debería especificarse algún requisito con más detalle?	2/4/2010
4	Especificación de requerimiento.	¿El número de páginas que aparece definido en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?	2/4/2010
5	Especificación de requerimiento.	¿Se han identificado los requerimientos de software y de hardware?	2/4/2010
6	Especificación de requerimiento.	¿Los requerimientos de soporte y usabilidad se han identificado?	2/4/2010
7	Especificación de requerimiento.	¿Han sido identificadas las restricciones de la implementación?	2/4/2010

Anexo 3 Lista de chequeo aplicada al sistema.

Seguridad

No	Importancia	Parámetros	Evaluación
1	Alta	¿El sistema cuenta con un proceso de autenticación?	5
2	Alta	¿Todos los usuarios tienen asignados un rol en el sistema?	5
3	Alta	¿Todos los roles tienen definidos niveles de acceso en al sistema?	5

Usabilidad

No	Importancia	Parámetros	Evaluación
1	Media	¿Es simple el vocabulario utilizado en el sistema?	5
2	Alta	¿El sistema es fácil de manejar para usuarios con poca experiencia en el trabajo con computadoras?	4
3	Alta	¿El sistema cuenta con una ayuda?	1
4	Media	¿Se entienden la interfaz y su contenido?	4
5	Alta	¿El sistema cuenta con un Manual de Usuario como ayuda para trabajar con la aplicación?	1
6	Alta	¿Actúa el sistema en la información de los errores?	5
7	Media	¿Se presenta al usuario la información que sólo necesita?	5

Portabilidad

No	Importancia	Parámetros	Evaluación
1	Media	¿Existe independencia del ambiente de hardware? (características particulares del hardware)	5
2	Media	¿Existe independencia del ambiente de software? (sistema operativo, lenguaje de programación)	5

Confiabilidad

No	Importancia	Parámetros	Evaluación
1	Media	¿Se recupera el software ante fallas?	5
4	Alta	¿El sistema garantiza su funcionamiento durante las 24 horas del día y los 7 días de la semana?	5

Glosario de Términos

Stakeholders: Individuos pueden afectar o son afectados por las actividades de una empresa.

Bucles: Ciclos

Casos de prueba: Productos de desarrollo software que ayudan a validar y verificar las expectativas de los Stakeholders.

Listas de Chequeo: Guía que permite al probador detectar errores existentes en la documentación.

No Conformidades: Errores encontrados durante la realización de las pruebas a un sistema informático.