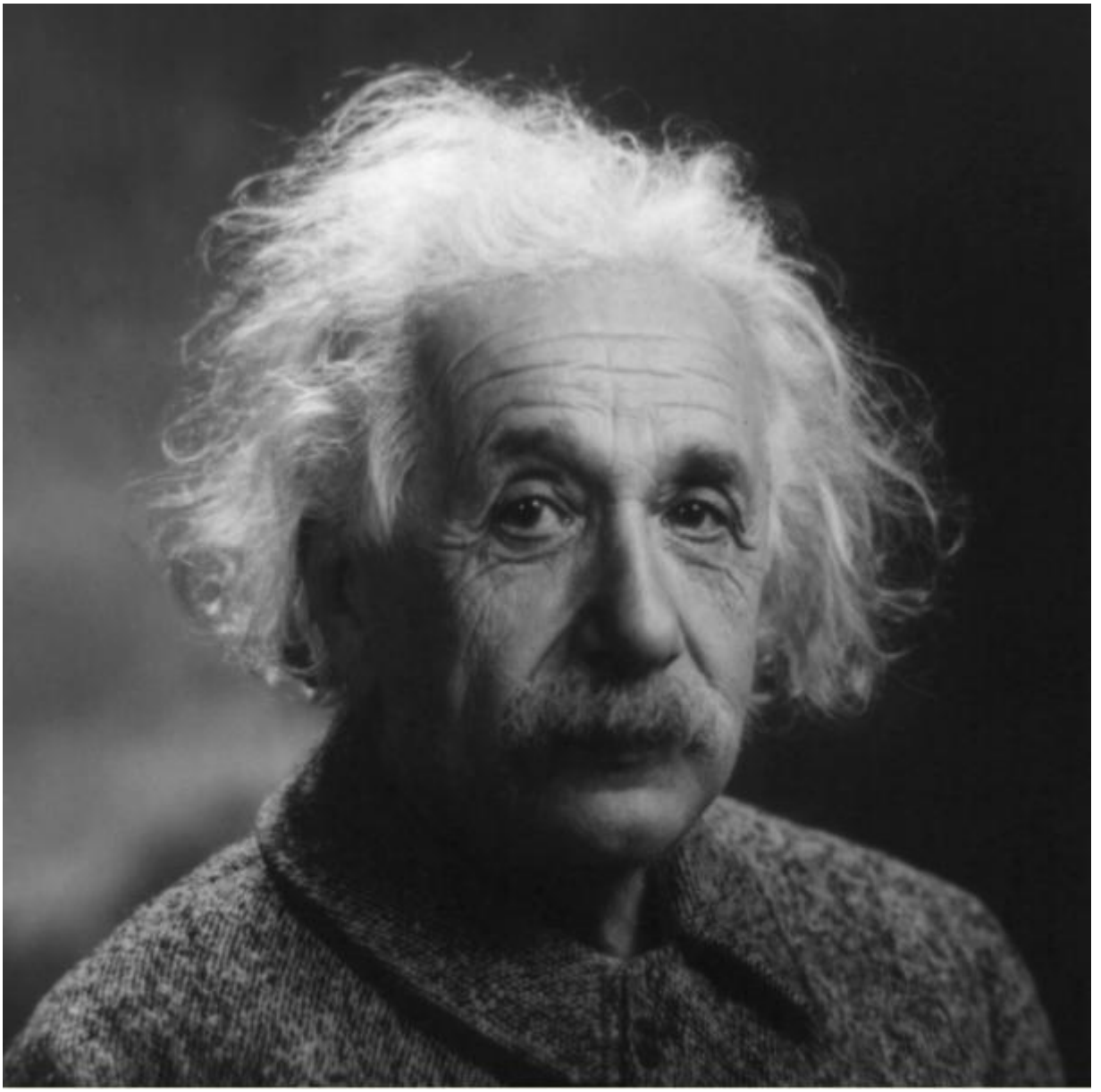
Universidad de las Ciencias Informáticas
Facultad 9

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

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

Autor: Bradier González Méndez
Tutor: Ing. Jesse Daniel Cano Otero.

Ciudad de la Habana, Mayo 2010.



“La formulación de un problema es más importante que su solución.”

Albert Einstein

Agradecimientos

A mi madre Maritza Méndez Hernández, por ser una de las personas impulsoras en mi formación profesional, la principal razón de mi existencia y mi ángel guardián durante toda mi vida.

A mi padre Andrés González Cortiñas, mientras estuvo vivo nunca faltaron sus sabios consejos, por sus deseos de verme convertido en ingeniero y por regañarme en momentos cruciales de mi vida.

Gracias papá!!!

A mis abuelos María Hernández Rodríguez y Roberto Méndez Arencibia, pilares fundamentales en mi niñez, por su apoyo, cariño y por llegar a vivir tantos años para verme convertido en un profesional.

A Osciris(papito) e Idalia, mis representantes y guías durante muchos años, por su preocupación e interés durante toda la carrera y por ser mis tutores paternos.

A mi hermano Vilier González Méndez, por su amor inigualable de hermano.

A mi hermana Yázara Gallardo Méndez, por ser mi única hermanita, por ser la guardiana de nuestra madre durante muchos años.

A Enmanuel Abelenda Rodríguez, mi compañero de estudio, de aula y mi amigo fiel.

A todo el piquete de la POLVASA, Sergio (La Mákina), Luisito (Luiso Bare Streisend), Enmanuel (Potaje), Jesse (El Insasiable), Marcelo (Y un besito pa mi mamá), Camilo (El Turista), Marlon, Diana (La Indomable), Lianet (La Zombi), Mirusleidys (Chuncha la campista), Daniela (Mi Reina).

Dedicatoria

A mis padres:
Maritza Méndez Hernández
Andrés González Cortiñas
A mis abuelos:
María Hernández Rodríguez
Roberto Méndez Arencibia
A mis hermanos:
Vilier González Méndez
Yázara Gallardo Méndez
A mi novia:
Daniela Borges Crosa
A mi hermano de crianza:
Yoicy Martín Cabrera
A mis grandes amigos de verdad:
Lizandro Beltrán
Máximo Briso Fernández
A todos mis otros amigos.

Declaración de autoría

Declaro que soy el único autor 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_____.

Autor: Bradier González Méndez

_____

Tutor: Ing. Jesse Daniel Cano Otero

_____

Resumen

Propiciar la calidad en el Software es una actividad que ha surgido como consecuencia de la fuerte demanda de sistemas de software en todos los procesos que se desarrollan en la actualidad; desde programas elementales de contabilidad hasta programas complejos como los espaciales. De allí el esfuerzo que se ha desplegado para obtener software de alta calidad. El aseguramiento de la calidad toma en cuenta todas aquellas acciones planificadas y sistemáticas necesarias para proporcionar la confianza de que un producto o servicio satisface los requisitos de calidad establecidos. Para que sea efectivo, se requiere una evaluación permanente de aquellos factores que influyen en la adecuación del diseño y de las especificaciones según las aplicaciones previstas, además la garantía de calidad del software es una “actividad de protección” que se aplica a lo largo de todo el proceso de Ingeniería del Software, la cual engloba: métodos y herramientas de análisis, diseño, codificación y prueba; revisión de técnicas formales que se aplican durante cada paso; control de la documentación del software y de los cambios realizados; procedimientos que aseguren un ajuste a los estándares de desarrollo del software; y mecanismos de medida y de información.

En el presente trabajo, se realiza un estudio de las técnicas de prueba con el objetivo de desarrollar una iteración del proceso de prueba sobre el módulo “Inventario de Minerales Sólidos” del proyecto Sistema de Gestión de Datos Geológicos, de manera que dicho módulo pueda ser entregado libre de defectos.

Palabras Claves

Calidad de Software, Casos de Pruebas, Estrategia de Pruebas, Plan de Pruebas, Pruebas de Software.

ÍNDICE DE FIGURAS

Índice de Figuras

ÍNDICE DE TABLAS

TABLA DE CONTENIDO

Tabla de Contenido

Introducción

En las décadas de, aproximadamente, 1960-1970 cuando comenzaba el auge de las computadoras y el interés por estas se incrementaba de forma acelerada, el gasto fundamental era en el hardware, y el software era gratuito o se incluía con el hardware. Posteriormente, por un desarrollo eminente, la llegada de la red global y el progresivo auge en las tecnologías de la informática y las comunicaciones, despertaron un excesivo interés tanto en programadores como en múltiples compañías, acontecimientos que fueron suficientes para que el software dejara de ser una herramienta de poco aval y convertirse así en una industria de gran importancia formando un pilar fundamental para el avance y la revolución de la esfera.

En los últimos años el desarrollo de la tecnología ha ido creciendo de forma apresurada y dentro de ello, el software ha sido uno de los elementos que ha formado parte de este severo impulso, convirtiéndose así en un tema de suprema importancia. En la sociedad actual, permanece la insatisfacción de los productos, o la continua solicitud en el cambio de estos por productos, mucho más avanzados, que abarquen la satisfacción de otras necesidades, que tengan una calidad mayor y costos más aceptables, lo que ha propiciado un enfoque fundamental en el proceso de desarrollo del software con vistas al mejoramiento de su calidad. Las grandes empresas enfrascadas en el desarrollo del software intentan constantemente perfeccionar la calidad de sus producciones no solo para saciar la insatisfacción de sus clientes, sino para entregar los productos en el plazo establecido y que cumplan las funciones requeridas y sobre todo para poder entrar o mantenerse en un mercado altamente competitivo, para lo cual se necesitan producciones de suma calidad.

Teniendo en cuenta que la demanda crece de forma apresurada, que esta industria es uno de los renglones fundamentales de ingresos económicos de muchos países en desarrollo y que pudiera ser una vía emergente para el embargo económico, además considerando la importancia que ha adquirido para el desarrollo en las diferentes ramas de la sociedad y teniendo en cuenta que la producción de sistemas y herramientas informáticas provocan el surgimiento de nuevas profesiones y mercados, además de ser una de las principales industrias del futuro, y una alternativa de respuesta a la situación económica actual, en nuestro país se ha despertado un gran interés por este mercado. Por lo que Cuba ha venido desarrollando proyectos que sean capaces de cumplir con características productivas y competitivas no solo en el ámbito nacional sino a nivel internacional también.

La Universidad de las Ciencias Informáticas (UCI) fue creada con el objetivo de producir software altamente calificado, no solo para cubrir demandas nacionales sino para su comercialización a nivel mundial. De esta forma, dentro de dicha universidad se llevan a cabo numerosos proyectos, los cuales abarcan áreas muy

variadas y para que la producción de estos se realicen al máximo nivel de la profesión y con la calidad requerida, se hace necesario y de vital importancia una estrategia de pruebas o variante de estas, las cuales garanticen la máxima calidad de cada producto. La realización de pruebas es una tarea que debe realizarse en cada sistema, pues permiten comprobar el grado de cumplimiento respecto a las especificaciones en el inicio del sistema, además pueden servir para diferentes propósitos:

1 Pueden demostrar que el sistema cumple los criterios de rendimiento.

2 Pueden comparar dos sistemas para ver cuál de ellos funciona mejor.

3 Pueden medir qué partes del sistema funcionan incorrectamente.

También se debe tener en cuenta que el proceso de pruebas se debe realizar durante todo el ciclo de desarrollo y no simplemente al final, porque pudieran detectarse errores al final de la etapa de desarrollo y darle solución se haría sumamente complejo. Debido a esto, es aconsejable que la realización de las pruebas pertinentes se lleve a cabo siguiendo una estrategia que cumpla los requisitos suficientes para validar cualquier error existente en el sistema.

Uno de los productos que en estos momentos se desarrolla dentro de la universidad es el Sistema de Gestión de Datos Geológicos (SDGD), el cual está formado por 8 módulos, los cuales tienen funciones diferentes, uno de ellos es el de Inventario de Minerales Sólidos, que tiene como objetivo administrar y proveer toda la información sobre el estado de los recursos y las reservas minerales, controlando y garantizando el uso racional de los mismos. En él se gestiona la información referida a los depósitos, sectores, unidades, leyes de corte y parámetros de calidad.

A pesar de encontrarse en desarrollo por un equipo calificado, la existencia de errores no es nula, lo que hace necesario buscar, encontrar, e informar dichos errores en el módulo Inventario de Minerales Sólidos y cuando el sistema esté concluido se hace aún más importante la revisión del mismo con el objetivo de demostrar que cumple con sus requerimientos y funciona según su diseño, siendo de vital importancia llevar a cabo un proceso de pruebas, y así de esta forma encontrar cualquier defecto existente o informar que el producto se encuentra en perfectas condiciones, por lo que puede ser entregado al cliente para su explotación.

El sistema está en desarrollo para la Oficina Nacional de Recursos Minerales (ONRM), la cual:

- Mediante el Acuerdo No. 5479, del Comité Ejecutivo del Consejo de Ministros, de fecha 7 de junio del 2005, referente a la reorganización del Ministerio de la Industria Básica, se dispone aprobar con carácter provisional, hasta tanto sea adoptada la nueva legislación sobre la Organización de la

Administración Central del Estado, los objetivos, funciones y atribuciones de dicho organismo, encontrándose entre éstas, las de dirigir, ejecutar y controlar la política del estado y del gobierno en cuanto a las actividades de búsqueda, exploración, explotación y procesamiento de minerales sólidos; así como la de mantener y actualizar el Balance Nacional de Recursos y Reservas.

- Mediante la ley 76 de Minas, promulgada el 23 de enero de 1995, en su artículo 14 incisos b) y f) establece que corresponde a la Oficina Nacional de Recursos Minerales, en su condición de autoridad minera, aprobar, registrar y controlar las reservas minerales, certificando el grado de preparación de los yacimientos para su asimilación industrial, así como constituirse en depositario de la información geológica y minera de la nación y mantener actualizadas las estadísticas mineras del país.

- Mediante el Acuerdo No. 3985, de 17 de abril del 2001, del Comité Ejecutivo del Consejo de Ministros, le fue encomendada a la Oficina Nacional de Recursos Minerales, en lo adelante la ONRM, entre otras funciones y atribuciones, la de aprobar los cálculos de reservas de los minerales sólidos, líquidos y gaseosos y mantenerlos actualizados.

- Se hace necesario actualizar la Clasificación de los Recursos y Reservas de Minerales Sólidos atendiendo a la tendencia mundial y a las particularidades nacionales y establecer una guía general para la clasificación, estimación y control de los recursos de minerales sólidos y los objetivos del Balance Nacional de dichos recursos y reservas de la República de Cuba; por lo que resulta conveniente derogar la Resolución No. 215, de 19 de julio de 1999, del Ministro de la Industria Básica, por medio de la cual se aprobó y puso en vigor la “Clasificación de Recursos y Reservas de Minerales Útiles Sólidos”.

Según los planteamientos anteriores queda como ***problema a resolver*** ¿Cómo lograr la entrega del módulo Inventario de Minerales Sólidos libre de defectos?

Para darle solución se define como ***Objeto de Estudio****:* El proceso de pruebas de sistemas de gestión de información.

Siendo el ***Campo de Acción:*** Diseño e implementación de las pruebas del módulo Inventario de Minerales Sólidos.

Para resolver el problema se debe dar cumplimiento al ***Objetivo General*** : Elaborar la documentación técnica del proceso de pruebas correspondiente al módulo Inventario de Minerales Sólidos que forma parte del Sistema de Gestión de Datos Geológicos (SGDG).
Obteniendo como posibles resultados: La evaluación y validación de la implementación del módulo Inventario de Minerales Sólidos a partir del desarrollo del proceso de pruebas.

El diseño y la implementación de las pruebas al módulo Inventario de Minerales Sólidos que forma parte del Sistema de Gestión de Datos Geológicos (SGDG), garantizarán que el mismo se entregue libre de defectos, siendo este argumento la ***Idea a Defender.***

Uno de los pasos importantes no es sólo definir el objetivo general de la investigación sino darle solución y para ello se hace necesario emplear métodos científicos, como los **Métodos Teóricos** y los **Métodos Empíricos.**
Los Métodos Teóricos: En estos están comprendidos toda una serie de procedimiento que posibilitan la asimilación teórica de la realidad y que se adecuan a las condiciones en que se va a desarrollar la investigación, además posibilitan el conocimiento del estado del arte del fenómeno, su evolución en una etapa determinada y su relación con otros fenómenos. **(1)**
Los métodos utilizados en la investigación:
Método Analítico - Sintético: Se utiliza para buscar específicamente las características de los defectos encontrados durante todo el proceso de pruebas realizado al módulo para poder darles solución.

Método Histórico – Lógico: Se usó para poder investigar y adquirir toda la información que se tiene hasta el momento sobre las pruebas.

Método de Modelación: Se usó para modelar los casos de prueba, pues es una necesidad que se ejecute esta modelación a la hora de aplicar una prueba, este método se pone en práctica para poder estudiar los diagramas elaborados por los analistas del equipo de desarrollo.

Los Métodos Empíricos: Permiten extraer de los fenómenos analizados, la información que se necesita sobre ellos a través de observaciones. **(1)**

Método de Observación: Es un método muy utilizado o generalmente utilizado en todas las investigaciones, donde se puede adquirir información para desarrollar una determinada investigación tomando como criterios y posibles soluciones las ya planteadas y elaboradas por otros autores. Además cuando se debe trabajar con los diagramas diseñados en conjunto con el software desarrollado para tomar decisiones pertinentes a la investigación.

Estructura del Desarrollo de la Investigación

El contenido de este trabajo se encuentra estructurado en tres capítulos, los que se definen de la siguiente manera:

- **Capítulo 1:** "Fundamentación Teórica". En este capítulo se aborda fundamentalmente el estado actual de las actividades relacionadas con el Proceso de Pruebas. Además se realiza un estudio sobre metodologías y conceptos fundamentales del proceso.

- **Capítulo 2:** "Diseño y Aplicación de las Pruebas". Se elaboran los artefactos propuestos por RUP y requeridos (Plan de Pruebas, Estrategia de Pruebas y Casos de Prueba) para la aplicación de las pruebas de software al módulo Inventario de Minerales Sólidos que pertenece al Sistema de Gestión de Datos Geológicos (SGDG).

- **Capítulo 3**: "Resultados". Se hace un análisis de los resultados obtenidos después de haber realizado las pruebas pertinentes al módulo Inventario de Minerales Sólidos del proyecto Sistema de Gestión de Datos Geológicos y un resumen de los principales errores encontrados durante el proceso de pruebas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1. Introducción

El análisis bibliográfico permite profundizar en los aspectos investigativos, así como conocer los distintos criterios y valoraciones que tienen diferentes autores sobre una temática determinada. Este análisis resulta de gran importancia, pues constituye una base sólida que fundamenta los métodos, procedimientos, técnicas y conceptos manejados en la investigación.

En el presente capítulo se realiza una investigación sobre el estado del arte de las pruebas de software, un análisis de los conceptos de prueba desde el punto de vista de diferentes autores, así como un estudio de las diferentes metodologías para mejorar el proceso de pruebas. Se detallan los distintos tipos y técnicas de pruebas, y las características de una serie de herramientas para el apoyo de los especialistas en la administración y confección de artefactos de pruebas. Todo esto para desarrollar un adecuado proceso de pruebas que permita encontrar la mayor cantidad de errores posibles en el módulo Inventario de Minerales Sólidos del proyecto “Sistema de Gestión de Datos Geológicos”.

1.2. Pruebas de Software

Las **pruebas de software**, son los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas. **(18)**

Para determinar el nivel de calidad se deben efectuar unas medidas o pruebas que permitan comprobar el grado de cumplimiento respecto de las especificaciones en el inicio del sistema.

Hay muchos planteamientos a la hora de abordar el proceso de pruebas de software, pero para verificar productos complejos de forma efectiva requiere de un proceso de investigación más que de seguir un procedimiento. Una práctica común es que el proceso de pruebas de un programa sea realizado por un grupo independiente de probadores al finalizar su desarrollo y antes de sacarlo al mercado. Una práctica que viene siendo muy popular es distribuir de forma gratuita una versión no final del producto para que sean los propios consumidores los que la prueben.

Finalmente y antes de salir al mercado, es cada vez más habitual que se realice una fase de “RTM (Release To Market) testing” (pruebas de liberación para el mercado), donde se comprueba cada funcionalidad del programa completo en entornos de producción.

Otra práctica es que el proceso de pruebas se realice desde el mismo momento en que empieza el desarrollo y continúe hasta que finaliza. **(2)**

El estándar ISO/IEC 12207 (ISO/IEC 1995) identifica tres grupos de procesos en el ciclo de vida software:

- Procesos principales, grupo en el que incluye los procesos de Adquisición, Suministro, Desarrollo, Operación y Mantenimiento. **(3)**

- Procesos de la organización, donde se encuentran los procesos de Gestión, Mejora, Infraestructura y Formación. **(3)**

- Procesos de soporte o auxiliares, donde están los procesos de Documentación, Gestión de la Configuración, Auditoria, Resolución de Problemas, Revisión Conjunta, Aseguramiento de la Calidad, Verificación, Validación. **(3)**

En los procesos de soporte o auxiliares encontramos dos procesos muy importantes:

El proceso de Validación, que tiene como objetivo determinar si los requisitos y el sistema final cumplen los objetivos para los que se construyó el producto.

El proceso de Verificación, que intenta determinar si los productos software de una actividad se ajustan a los requisitos o a las condiciones impuestas en actividades anteriores. **(3)**

1.2.1. Tipos de Prueba

1.2.1.1. Pruebas de Unidad

Son los procedimientos de pruebas locales a un módulo del sistema. Por definición dichas pruebas cubren la funcionalidad propia del módulo tanto con una perspectiva de caja blanca como de caja negra; pero prestando poca o ninguna atención a la integración con otros módulos.

Las pruebas de unidad están orientadas principalmente a validar el cumplimiento de los estándares de presentación y demás características visuales de la aplicación como la salida de los reportes. La prueba de unidad se centra en el módulo, usando la descripción del diseño detallado como guía, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo. Estas pruebas son comprobaciones que hacemos a las unidades lógicas de nuestro programa. Verificando que una unidad funcione correctamente por sí misma, sin tener en cuenta las relaciones que pueda tener con otras partes del sistema. Estas pruebas son pequeños módulos auxiliares, que se encargan de verificar el funcionamiento de otras unidades lógicas del sistema. Las unidades lógicas de un programa son aquellas partes en que lo hemos dividido para entenderlo mejor. Pueden ser los módulos, paquetes, clases, subsistemas, funciones o cualquier otro mecanismo que nos ofrezca el lenguaje de programación que estamos utilizando. **(17)**

1.2.1.2. Prueba de Caja Negra

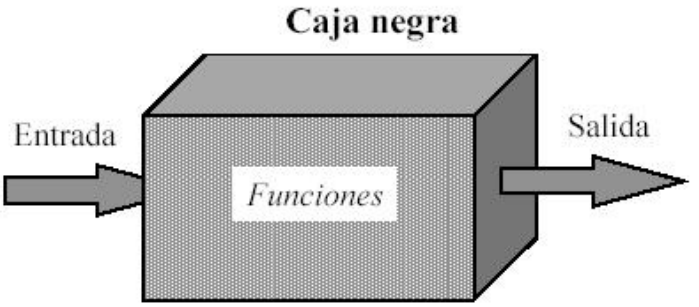


Figura 1 Enfoque de diseño de pruebas de caja negra.

Se centra en los requisitos funcionales del software. Permite obtener un conjunto de condiciones de entrada que ejerciten completamente los requisitos funcionales del programa. No es una alternativa a la prueba de caja blanca. Es un enfoque complementario. **(4)**

Las Pruebas de caja negra intentan hallar errores tales como:

1- Funciones incorrectas o ausentes.
2- Errores de interfaz.
3- Errores en estructuras de datos o en accesos a BD externas.
4- Errores de rendimiento.
5- Errores de inicialización o terminación.

Para desarrollar la prueba de Caja Negra existen varias técnicas, entre ellas están:

1- Técnica de la Partición de Equivalencia: Presenta la partición equivalente como un método de caja negra que divide el dominio de entrada de un programa en un conjunto de clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. El objetivo de partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases. **(5)**

En el diseño de casos de prueba para partición equivalente se procede en dos pasos:

- Se identifican las clases de equivalencia. Las clases de equivalencia son identificadas tomando cada condición de entrada (generalmente una oración o una frase en la especificación) y repartiéndola en dos o más grupos.

Es de notar que dos tipos de clases de equivalencia están identificados: las clases de equivalencia válidas representan entradas permitidas al programa, y las clases de equivalencia inválidas que representan el resto de los estados posibles de la condición (es decir, valores erróneos de la entrada).

- Se definen los casos de prueba. El segundo paso es el uso de las clases de equivalencia para identificar los casos de prueba. El proceso es como sigue: se asigna un número único a cada clase de

equivalencia. Hasta que todas las clases de equivalencia válidas han sido cubiertas por los casos de prueba, se escribe un nuevo caso de prueba que cubra la clase de equivalencia válida. Y por último hasta que los casos de prueba hayan cubierto todas las clases de equivalencia inválidas, se escribe un caso de la prueba que cubra una, y solamente una, de las clases de equivalencia inválidas descubiertas**. (6)**

2- Análisis de valores límite (AVL): El análisis de valores límite es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida. **(5)**

Condiciones sublímite*.* Las condiciones límite normales son las más obvias de descubrir. Estas son definidas en la especificación o son evidentes al momento de utilizar el software. Algunos límites, sin embargo, son internos al software, no son necesariamente aparentes al usuario final pero aún así deben ser probadas por el probador. Estas son conocidas como condiciones sublímites o condiciones límite internas. Una condición sublímite común es la tabla de caracteres ASCII, por ejemplo, si se está evaluando una caja de texto que acepta solamente los caracteres AZ y az, se deben incluir los valores en la partición inválida justo «debajo de» y «encima de» esos caracteres de la tabla ASCII. **(7)**

3- Métodos de prueba basados en grafos: En este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. **(5)**

En este método:

- Se crea un grafo de objetos importantes y sus relaciones.
- Se diseñan una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores.

4- Adivinando el error: Dado un programa particular, se conjetura, por la intuición y la experiencia, ciertos tipos probables de errores y entonces se escriben casos de prueba para exponer esos errores. Es difícil dar un procedimiento para esta técnica puesto que es en gran parte un proceso intuitivo. **(6)**

1.2.1.3. Prueba de Caja Blanca

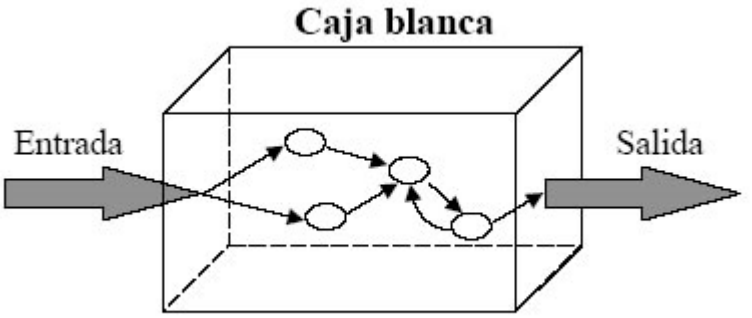


Figura 2 Enfoque de diseño de pruebas de caja blanca.

La **prueba de caja blanca** denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba.

Mediante este método el ingeniero de software puede obtener casos de prueba que:

1. Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
2. Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites y con sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez. **(5)**

En el libro “Software Testing” (“Pruebas de Software”, traducido al español), el autor Ron Patton presenta dos términos que describen la forma de realizar pruebas: *Pruebas Estáticas* y *Pruebas Dinámicas*. Para el autor estos términos permiten definir dos clases de pruebas de caja blanca:

- **Pruebas estáticas de caja blanca**: Es el proceso que cuidadosamente y metódicamente revisa el diseño del software, la arquitectura o el código para encontrar defectos sin necesidad de ejecutar el código. Esto algunas veces se refiere a un análisis estructural.
- **Pruebas dinámicas de caja blanca**: En estas pruebas se revisa dentro de la caja, se examina el código y se observa éste mientras se ejecuta. La prueba de caja blanca dinámica, en resumidas palabras, utiliza la información que se obtiene al observar qué hace el código, cómo trabaja, para así determinar qué probar, qué no probar y cómo aproximarse a las pruebas.

Algunos de los métodos empleados en las pruebas de caja blanca son los siguientes:

1- Prueba del camino básico: Es una técnica propuesta inicialmente por Tom McCabe , la cual le permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizarán que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. **(8)**

De forma general, los pasos que se deben seguir para la obtención de los casos de prueba en este método, son los siguientes:

1. Emplear el diseño o el código para elaborar el grafo de flujo.
2. Determinar la complejidad ciclomática del grafo de flujo.
3. Determinar un conjunto básico de caminos linealmente independientes.
4. Preparar los casos de prueba que forzarán la ejecución de cada camino del conjunto básico. **(5)**

2- Prueba de condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa. **(9)** Algunos conceptos empleados alrededor de esta prueba son los siguientes:

- Condición simple*:* es una variable lógica o una expresión relacional (E_1 < *operador - relacional* > E_2).
- Condición compuesta*:* está formada por dos o más condiciones simples, operadores lógicos y paréntesis.

En general los tipos de errores que se buscan en una prueba de condición, son los siguientes:

- Error en operador lógico (existencia de operadores lógicos incorrectos, desaparecidos, sobrantes).
- Error en variable lógica.
- Error en paréntesis lógico.
- Error en operador relacional.
- Error en expresión aritmética.

3- Prueba del flujo de datos: Selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. **(9)**

4- Prueba de bucles: Es una técnica que se centra exclusivamente en la validez de las construcciones de bucles (bucles simples, anidados, concatenados y no estructurados).

Bucles simples*:* Se les aplica el siguiente conjunto de pruebas:

- Pasar por alto totalmente el bucle.
- Pasar una sola vez por el bucle.
- Pasar dos veces por el bucle.

Bucles anidados*:* Si se empleara el mismo enfoque de prueba de bucles simples a los bucles anidados, el número de pruebas aumentaría considerablemente por lo cual se sugiere emplear el siguiente enfoque:

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

- Continuar hasta que se hayan probado todos los bucles. **(19)**

Bucles concatenados*:* Estos bucles se pueden probar utilizando el enfoque de bucles simples, siempre y cuando cada uno de los bucles sea independiente del resto de lo contrario se debe emplear el enfoque de bucles anidados.

Bucles no estructurados*:* Siempre que sea posible estos bucles deben rediseñarse.

1.2.1.4. Prueba de Integración

Es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación. Estas pruebas descubren errores en la implementación de los paquetes y pueden indentificar que las especificaciones de las interfaces de estos están incompletas. Esta prueba debe ser responsabilidad de desarrolladores y de probadores independientes, sin que se solapen las pruebas efectuados por estos partidos. Es el proceso de combinar y probar múltiples componentes juntos. El objetivo es tomar los componentes probados en unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.

Se llama integración incremental cuando el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir, es más probable que se puedan probar completamente las interfaces y se puede aplicar un enfoque de prueba sistemática.

Hay dos estrategias de integración incremental **(20)**

Integración Descendente (Top-Down):

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

Este proceso se realiza en una serie de cinco pasos:

1. Se usa el módulo de control principal como controlador de la prueba, disponiendo de resguardos para todos los módulos directamente subordinados al módulo de control principal.
2. Dependiendo del enfoque de integración elegido se van sustituyendo los resguardos subordinados uno a uno por los módulos reales.
3. Se llevan a cabo pruebas cada vez que se integra un nuevo módulo.

4. Tras terminar cada conjunto de pruebas, se reemplaza otro resguardo con el módulo real.
5. Se hace la prueba de regresión para asegurarse de que no existen errores nuevos.

El programa continúa desde el paso 2 hasta que se haya construido la estructura del programa entero. Al aplicar esta estrategia pueden surgir algunos problemas, el más común se da cuando se requiere un proceso de los niveles más bajos de la jerarquía para poder probar adecuadamente los niveles superiores. Al principio de la prueba descendente, los módulos de bajo nivel se reemplazan por resguardos; por tanto, no pueden fluir datos significativos hacia arriba por la estructura del programa. Para solucionar esto se tienen tres opciones:

- Retrasar muchas de las pruebas hasta que los resguardos sean reemplazados por los módulos reales.
- Desarrollar resguardos que realicen funciones limitadas que simulen los módulos reales.
- Integrar el software desde el fondo de la jerarquía hacia arriba. **(20)**

Integración Ascendente (Bottom-Up):

Se refiere a la identificación de aquellos procesos que necesitan computarizarse conforme vayan apareciendo, su análisis como sistema y su codificación, o bien, la adquisición de paquetes de software para satisfacer el problema inmediato.

Se puede implementar una estrategia de integración ascendente mediante los siguientes pasos:

1. Se combinan los módulos de bajo nivel en grupos que realicen una subfunción específica del software.
2. Se escribe un controlador para coordinar la entrada y la salida de los casos de prueba.
3. Se prueba el grupo.
4. Se eliminan los controladores y se combinan los grupos moviéndose hacia arriba por la estructura del programa.

A medida que la integración progresa disminuye la necesidad de controladores de prueba diferentes. La selección de una estrategia de integración depende de las características del software y de la planificación del proyecto.

Una buena alternativa es usar una mezcla de las dos estrategias (Ascendente y Descendente) que use la descendente para los niveles superiores de la estructura, junto con la ascendente para los niveles subordinados.

A medida que progresa la prueba de integración, se deben identificar los módulos críticos. Un módulo crítico es aquel que tiene una o más de las siguientes características:

- Está dirigido a varios requisitos del software
- Tiene un mayor nivel de control
- Es complejo o propenso a errores
- Tiene unos requisitos de rendimiento muy definidos
- Los módulos críticos deben probarse lo antes posible. **(6)**

1.2.1.5. Pruebas de Validación:

Tiene como objetivo fundamental obtener información para la validación de los algoritmos. Se asume para esta parte que el software ha sobrepasado la etapa de verificación, por lo que está libre de errores en tiempo de ejecución, lo que no significa que esté libre de errores lógicos. **(11)**

El software totalmente ensamblado se prueba como un todo para comprobar si cumple los requisitos funcionales y de rendimiento, facilidad de mantenimiento, recuperación de errores, etc.

El objetivo de estas pruebas es obtener información útil para la validación de la implementación de los algoritmos implementados. Se asume para esta parte que el software ha cumplido la etapa de verificación, por lo tanto está libre de errores de tiempo de ejecución, lo que no significa que esté libre de errores lógicos.

1.2.1.6. Pruebas de Sistemas:

Tiene como objetivo fundamental verificar el sistema de software para ver si este cumple con sus requisitos. Dentro de esta fase pueden desarrollarse distintos tipos de pruebas, de las cuales algunas son funcionales, prueba de usabilidad, pruebas de rendimiento, pruebas de seguridad, etc. El trabajo es centrado fundamentalmente en pruebas funcionales de aplicaciones con interfaces gráficos, las cuales verifican que el software ofrece a los actores humanos la funcionalidad recogida en su especificación. **(12)**

Una de las técnicas más empleadas para la especificación funcional de sistemas software son los casos de uso. Las principales ventajas de los casos de uso son que ocultan los detalles internos del sistema, son rápidos de construir, fáciles de modificar y entender por los clientes y futuros usuarios del sistema **(13)** y pueden aplicarse a distintos tipos de sistemas **(15)**.

1.2.1.7. Tipos de Pruebas del Sistema

- *Prueba de Recuperación*: Es una prueba del sistema que fuerza el fallo del software de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente.**(5)**
- *Prueba de Seguridad*: Intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán de accesos impropios. **(5)**
- *Prueba de Resistencia*: Están diseñadas para enfrentar a los programas con situaciones anormales.
- *Prueba de Rendimiento*: Está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. **(20)**
- *Prueba de Funcionalidad*: Puede estar orientada a probar específicamente la función, que es cuando se fija la atención en la validación de las funciones, métodos, servicios, casos de uso entre otros, también puede tratar de buscar la seguridad asegurando que los datos o el sistema solamente son accedidos por actores deseados. O para probar el volumen, verificando las habilidades de los programas para manejar grandes cantidades de datos**. (20)**
- *Prueba de Stress*: Enfocadas a evaluar cómo el sistema responde bajo las condiciones anormales. Ejemplo: extrema sobrecarga, insuficiente memoria, servicios y hardware no disponibles o recursos compartidos no disponibles, entre otras**. (20)**

1.3. Niveles de Prueba.

La Prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Teniendo en cuenta esto, las pruebas se agrupan por niveles, de acuerdo con las diferentes etapas del proceso de desarrollo:

- *Prueba de desarrollador:* Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas han sido consideradas solo para la prueba de unidad, aunque en algunos casos pueden ejecutar pruebas de integración. Se recomienda que cubran más que las pruebas de unidad. **(24)**

- *Prueba independiente:* Es la prueba que es diseñada e implementada por alguien independiente del grupo de desarrolladores. El objetivo de estas pruebas es proporcionar una perspectiva diferente y en un ambiente más rico que los desarrolladores. Una vista de la prueba independiente es la prueba independiente de los stakeholder, que son pruebas basadas en las necesidades y preocupaciones de los stakeholders. **(24)**

- *Prueba de unidad:* Es la prueba enfocada a los elementos "testeables" más pequeños del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca. **(17)**

- *Prueba de integración:* Tienen por objetivo verificar el conjunto funcionamiento de dos o más módulos, si bien se deben poner en práctica desde la creación de dos módulos que interactúen entre sí, en el supuesto caso que se necesiten más de dos módulos para efectuar las pruebas, deberán generarse simples emuladores de módulos que entreguen datos esperados para la prueba individual de cada uno Es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación. Se diseñan para descubrir errores en las especificaciones de las interfaces de los paquetes y son responsabilidad de desarrolladores y de independientes. **(10)**

- *Prueba de sistema:* Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. **(6)**

- *Prueba de aceptación:* Prueba de aceptación del usuario es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido. **(24)**

Es importante tener en cuenta que las pruebas de unidad son implementadas en la iteración más temprana como el primer nivel de prueba. Pero en un proceso iterativo ejecutar todas las pruebas de unidad antes de

pasar a niveles siguientes de prueba como regla es inapropiada. Una mejor estrategia es identificar las pruebas de unidad, integración y sistema que ofrecen mayor potencial para encontrar errores y entonces implementarlas y ejecutarlas. **(17)**

1.4. Estrategia de Prueba.

Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la construcción correcta del software, los objetivos fundamentales para trazar una estrategia de prueba son:

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de unidad, integración y las pruebas de sistema. Las pruebas de unidad y de integración son necesarias dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.
- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, cómo realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas.
- Realizar diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Los productos de desarrollo de software en los que se detectan defectos son probados de nuevo y posiblemente devueltas a otra etapa, como diseño o implementación, de forma que los defectos puedan ser arreglados.

Para conseguir estos objetivos el flujo de trabajo de pruebas consta de las etapas de planificación, diseño, implementación ejecución y evaluación de las pruebas y además los productos de desarrollo del software fundamentales que se desarrollan en la etapa de pruebas son el plan de pruebas, los casos de prueba, el informe de evaluación de las pruebas, el modelo de prueba que incluye a su vez las clases de prueba, el entorno de configuración de pruebas, componentes de prueba y datos de las pruebas. **(25)**

1.5. Metodología de Desarrollo RUP

El Proceso Unificado de Desarrollo, habitualmente resumido como RUP, plantea quién hace qué, cuándo y cómo dentro del proceso de desarrollo de software, tiene como objetivos asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles, es dirigido por casos de uso, centrado en la

arquitectura, iterativo (mini-proyectos) e incremental (versiones), es actualizado constantemente para tener en cuenta las mejores prácticas y utiliza además UML como lenguaje de modelado.
La metodología RUP se divide en 4 fases el desarrollo del software:

- *Inicio:* El Objetivo en esta etapa es determinar la visión del proyecto.
- *Elaboración:* En esta etapa el objetivo es determinar la arquitectura óptima.
- *Construcción:* En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- *Transmisión:* El objetivo es llegar a obtener el release (liberación o lanzamiento) del proyecto.

Además las principales actividades para recorrer el ciclo de vida del software según RUP se encuentran repartidas en 6 flujos de trabajos ingenieriles:

- Modelamiento del Negocio.
- Requerimientos.
- Análisis y Diseño.
- Implementación.
- Pruebas.
- Despliegue.

Cada una de estas etapas es desarrollada mediante ciclos de iteraciones, lo cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. RUP aumenta la productividad de los desarrolladores mediante el acceso a la base de conocimiento, plantillas y herramientas, se centra en la producción y mantenimiento de modelos del sistema más que en producir documentos, pretende implementar las mejores prácticas actuales en la ingeniería de software teniendo en cuenta el desarrollo iterativo del software, la administración de requerimientos, el uso de arquitecturas basadas en componentes, el modelamiento visual del software, la verificación de la calidad del software y el control de cambios.

RUP describe cómo planear y ejecutar las pruebas teniendo en cuenta un grupo de artefactos, actividades y trabajadores. Dentro de los roles que propone se encuentra el de ingeniero de prueba el cual cumple con ciertas responsabilidades, por ejemplo, debe trabajar con el equipo de desarrollo para validar los requerimientos en aras de resolver las incidencias del diseño y los defectos del software, establecer los

juegos de prueba basándose en los requerimientos del cliente y el diseño, ejecutar planes de prueba en los entornos de integración y sistema de los nuevos release (nueva versión) y de regresión para asegurar la integridad de la aplicación, ejecutar los planes de prueba, dirigir el análisis detallado de los resultados de las pruebas tanto correspondientes a pruebas manuales como a las automáticas, colaborar con el equipo de trabajo para rectificar los errores encontrados, llevar a cabo pruebas de integración, funcionales, de aceptación y de interfaz de usuario, debe tener además la capacidad para revisar los requerimientos del sistema en busca de inconsistencias, contradicciones u omisiones críticas; así mismo, debe ser capaz de definir conjuntos de casos de prueba que permitan validar, con cierto nivel de confianza, que el requerimiento se satisface con la implementación propuesta; también, debe ser capaz de implementar aquellos programas para configurar el ambiente de pruebas y ejecutar de la manera más eficiente dichas pruebas. **(21)**

1.5.1. Artefactos del flujo de trabajo de prueba.

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

- *Artefacto modelo de prueba.*

Describe fundamentalmente cómo se prueban los componentes en el modelo de implementación ejecutando pruebas de integración y de sistemas, este artefacto puede describir también cómo han de ser probados aspectos específicos del sistema, por ejemplo, si la interfaz de usuario es utilizable y consistente o si el manual de usuario del sistema cumple con su cometido.

- *Artefacto caso de prueba.*

Especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se debe probar y las condiciones en las que debe probarse.

- *Artefacto componente de prueba.*

Automatizan uno o varios procedimientos de pruebas o partes de ellos, estos componentes pueden ser desarrollados utilizando un lenguaje de guiones o un lenguaje de programación, o pueden ser grabados con una herramienta de automatización de pruebas, se utilizan además para probar los componentes en el modelo de implementación, proporcionando entradas de pruebas, controlando y monitoreando la ejecución de los componentes a probar.

- *Artefacto procedimiento de prueba.*

Especifica cómo realizar uno o varios casos de prueba o partes de estos, por ejemplo 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 interactuar manualmente con una herramienta de automatización de pruebas para crear componentes ejecutables de ellas.

- *Artefacto plan de prueba.*

Describe las estrategias, recursos y planificación de las pruebas. La estrategia incluye la definición del tipo de prueba a realizar para cada iteración y sus objetivos, el nivel de cobertura, de código necesario y el porcentaje que debería ejecutarse con un resultado específico.

- *Artefacto defecto.*

Es una anomalía del sistema, como por ejemplo un síntoma de un fallo de software o un problema descubierto en una revisión. Un defecto puede ser utilizado para localizar cualquier cosa que los desarrolladores necesitan registrar como síntoma de un problema en el sistema que ellos necesitan controlar y resolver.

- *Artefacto evaluación de prueba.*

Es una evaluación de los resultados del esfuerzo de prueba, tales como la cobertura del caso de prueba, la cobertura del código y el estado de los defectos. **(22)**

1.5.2. Actividades fundamentales del flujo de trabajo de prueba.

Las actividades fundamentales que se desarrollan en el flujo de trabajo de pruebas son las siguientes:

- Planificar las pruebas.
- Diseñar las pruebas: Dentro de esta actividad se realizan pruebas de integración, de sistema y de regresión.
- Implementar prueba.
- Realizar pruebas de integración.
- Realizar pruebas de sistema.
- Evaluar pruebas. **(22)**

1.6. Conclusiones

En este capítulo se definieron conceptos fundamentales y estrategias a seguir en el proceso de pruebas del software, demostrando así la gran importancia de un desarrollo eficaz y llegando a la conclusión de los pasos específicos a seguir en cada prueba que se vaya a realizar, pretendiendo aportar información sobre la

realidad de las prácticas de las pruebas. Se decidió aplicar, al módulo Inventario de Minerales Sólidos, el tipo prueba de Caja Blanca con la técnica camino básico complementado con la prueba de Caja Negra con la técnica partición equivalente, la decisión estuvo basada en que estos tipos de pruebas son los recomendados para los niveles de prueba: Aceptación y Sistema que son los que se aplican en la fase en la que se encuentra el proyecto; además son los definidos por la dirección de calidad de la universidad para probar las liberaciones de los productos.

CAPÍTULO 2: DISEÑO Y APLICACIÓN DE LAS PRUEBAS

2.1. Introducción

El diseño y aplicación de las pruebas al módulo Inventario de Minerales Sólidos precisará una estrategia de prueba, la cual será la guía para todo el proceso que se llevará a cabo, tanto en la planificación, en el diseño de los casos de pruebas, la ejecución y los resultados.

Se realizarán pruebas específicas basándose en la documentación del software a probar y a partir de los resultados obtenidos, se pasará a su evaluación mediante comparación con la salida esperada.

2.2 Estrategias 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. **(2)**

2.2.1. Propósito

El propósito de la Estrategia de Prueba es el de analizar cómo plantear las pruebas en el ciclo de vida del **Módulo Inventario de Minerales Sólidos**, además integra un conjunto de actividades y técnicas de diseño de casos de pruebas en una serie de pasos bien planificados que dan como resultado la obtención de un producto final con calidad, además describe los pasos que se llevarán a cabo durante el proceso de pruebas.

2.2.2 Alcance

Las técnicas de prueba seleccionadas serán aplicadas sólo al **Módulo Inventario de Minerales Sólidos** con vista a aumentar la calidad del mismo, se realiza esta propuesta estratégica para plantear las pruebas que se desarrollarán. Se llevará a cabo la Prueba de Caja Blanca, la cual se centra en cada caso de uso del módulo individualmente asegurando su correcto funcionamiento. Las técnicas que más prevalecen son las de diseño de casos de prueba de Caja Negra, las cuales tendrán como objetivo comprobar si se cumplen o no los requisitos funcionales del módulo, así como la aceptación de forma adecuada de la entrada y salida de datos, teniendo en cuenta que son completamente indiferentes del comportamiento interno y de la estructura del programa.

2.3 Casos de Pruebas

Los casos de pruebas tienen como objetivo fundamental determinar si los requisitos de la aplicación son parcial o completamente satisfactorios a través de funciones determinadas. Estos casos de pruebas proporcionan una descripción de puntos importantes de observación, con pruebas positivas y negativas para lograr que la mayoría de los requisitos de la aplicación sean debidamente probados.

Las celdas de la tablas que se muestran a continuación pueden contener valores tales como: V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

2.4 Diseño de casos de pruebas de Caja Negra

Las pruebas de caja negra comprueban las funcionalidades del sistema. Con la aplicación de la técnica de partición equivalente en el módulo se podrá probar si los requerimientos planteados por el cliente en un inicio y descritos en casos de uso, fueron cumplidos en su conjunto sin fallos o errores.

2.4.1. Utilizando la Técnica Partición Equivalente

Para detallar el caso de uso se utiliza una tabla, donde se desglosa esta funcionalidad en secciones y a su vez estas en escenarios, para hacer más eficiente la ejecución de las pruebas.

A continuación se muestran los casos de prueba de caja negra para dos casos de uso críticos del módulo.

Nombre del CU: **Gestionar Depósito**
Actor: Administrador del Balance.
Descripción General: El caso de uso se inicia cuando el Administrador del Balance accede al menú Gestión y selecciona la opción Depósitos, el sistema le muestra las acciones a realizar y, en dependencia de las mismas, muestra el formulario donde debe introducir los datos necesarios para llevar a cabo la acción deseada. El CU termina una vez realizada la operación seleccionada y el Administrador del Balance sale de la aplicación.

Condiciones:
Precondiciones: El usuario debe estar autenticado y poseer los permisos de administración.

Poscondiciones: El sistema queda actualizado luego de realizada cualquier acción descrita anteriormente.

Tabla 1 Matriz Parcial de Escenarios del CU Gestionar Depósito

Nombre de la sección	ID Escenario	Nombre Escenario	Descripción de la Funcionalidad
SC 1: Seleccionar Opción	EC1	Mostrar Pestañas	Se muestran dos pestañas “Buscar” y “Adicionar” para la adición y búsqueda de los Depósitos.
SC 2: Adicionar Depósito	EC2	Adición Exitosa	El usuario selecciona la opción Adicionar Depósito, el sistema muestra un formulario que permite insertar un nuevo depósito con los campos: nombre, clasificación, provincia y municipios, el actor inserta todos los datos y da clic en el botón Enviar. El sistema valida que no exista un depósito con el mismo nombre, adiciona el nuevo depósito y muestra un mensaje señalando que la adición fue exitosa.
	EC3	Campos Vacíos.	El usuario selecciona la opción Adicionar Depósito, el sistema muestra un formulario que permite insertar un nuevo depósito con los campos: nombre, clasificación, provincia y municipios, el actor deja alguno de los campos en blanco y da clic en el botón Enviar. El sistema marca en rojo los campos en blanco y muestra un mensaje indicando que existen campos vacíos.
	EC4	Depósito Existente	El usuario selecciona la opción Adicionar Depósito, el sistema muestra un formulario que permite insertar un nuevo depósito con los campos: nombre, clasificación, provincia y municipios, el actor llena los datos pero en el campo nombre entra un nombre de depósito que ya existe y da clic en el botón Enviar. El sistema muestra un mensaje de error indicando que

			ya existe un depósito con ese nombre.
SC 3: Buscar Depósito	EC5	Buscar Depósito	El usuario selecciona la opción Buscar Depósito, luego inserta en el campo Buscar cualquier dato que contengan los campos nombre, clasificación, municipio, provincia. El sistema busca el o los depósitos de acuerdo a los datos insertados y los muestra en una tabla de la siguiente manera: nombre, clasificación, municipio, provincia, Acciones (Modificar, Eliminar, Ver Detalles) y muestra además la opción de generar un documento con los datos de la lista obtenida.
SC 4: Modificar Depósito	EC6	Modificación	El actor a partir del flujo de EC5 selecciona la opción Modificar de alguno de los depósitos mostrados. El sistema muestra una página que contiene un formulario para modificar los datos del depósito (aparecen los mismos campos que en el Adicionar Depósito). El usuario modifica los valores que desee y da clic en el botón Enviar. El sistema guarda los datos y muestra un mensaje indicando que la modificación se realizó con éxito.
SC 5: Eliminar Depósito	EC7	Eliminación Aceptada	El actor a partir del flujo de EC5 selecciona la opción Eliminar de alguno de los depósitos mostrados. El sistema muestra un mensaje solicitando la confirmación de la eliminación. El usuario da clic en el botón Aceptar y el sistema elimina el depósito y muestra un mensaje indicando que se procedió con la eliminación exitosamente.
	EC8	Eliminación Cancelada	El actor a partir del flujo de EC5 selecciona la opción Eliminar de alguno de los depósitos mostrados. El sistema muestra un mensaje solicitando la

			confirmación de la eliminación. El usuario da clic en el botón Cancelar y el sistema no elimina el depósito.
SC 6: Ver Detalles	EC9	Ver Detalles	El actor a partir del flujo de EC5 selecciona la opción Ver Detalles de alguno de los depósitos mostrados. El sistema muestra, en una ventana en forma de tabla, toda la información del depósito seleccionado.

Tabla 2 Matriz del CP del CU Gestionar Depósito .SC 1: Seleccionar Opción

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Mostrar pestañas.	El sistema muestra en una pantalla dos pestañas: "Buscar" y "Adicionar".		Principal Balance M Gestión Depósito

Tabla 3 Matriz del CP del CU Gestionar Depósito .SC 2: Adicionar Depósito

Escenario	Nombre del Depósito	Clasificación del Depósito	Provincia	Municipio	Campo Buscar	Respuesta del Sistema	Resultado de la prueba	Flujo Central
Adición Exitosa	V	V	V	V	N/A	El sistema adiciona el depósito y muestra un mensaje de confirmación "Se ha insertado un depósito".		Principal Balance M Gestión Depósitos Adicionar

Campos Vacíos	Algunos de los campos se dejan en blanco o se introducen datos de forma incorrecta.				N/A	El sistema selecciona en rojo los campos que no pueden quedar vacíos y muestra un mensaje: "Existen		Principal Balance M Gestión Depósitos Adicionar
Depósito Existente	I	V	V	V	N/A	El sistema muestra un mensaje de error porque existe otro sector con el mismo nombre: "Ya existe un depósito con ese nombre".		Principal Balance M Gestión Depósitos Adicionar

Tabla 4 Matriz del CP del CU Gestionar Depósito .SC 3: Buscar Depósito

Escenario	Nombre del Depósito	Clasificación del Depósito	Provincia	Municipio	Campo Buscar	Respuesta del Sistema	R.P	Flujo Central

Buscar Depósito	N/A	N/A	N/A	N/A	V	El sistema busca el o los depósitos de acuerdo a los datos insertados y los muestra en una tabla de la siguiente manera: nombre del depósito, municipio, provincia, Acciones (Modificar, Eliminar, Ver Detalles) y muestra además la opción de generar un documento con los datos de la lista obtenida.		Principal Balance M Gestión Depósitos Buscar

Tabla 5 Matriz del CP del CU Gestionar Depósito .SC 4: Modificar Depósito

Escenario	Nombre del Depósito	Clasificación del Depósito	Provincia	Municipio	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Modificación	N/A	N/A	N/A	N/A	V	El sistema modifica los datos del depósito y muestra un mensaje: “Depósito actualizado”.		Principal Balance M Gestión Depósitos Buscar Modificar

Tabla 6 Matriz del CP del CU Gestionar Depósito .SC 5: Eliminar Depósito

Escenarios	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central

Eliminación Aceptada	N/A	El sistema elimina el Depósito y actualiza la lista de estos.		Principal Balance M Gestión Depósitos Buscar Eliminar
Eliminación Cancelada	N/A	El sistema no elimina el Depósito.		Principal Balance M Gestión Depósitos Buscar Eliminar

Tabla 7 Matriz del CP del CU Gestionar Depósito .SC 6: Ver Detalles

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Ver Detalles	El sistema muestra, en una ventana en forma de tabla, toda la información del depósito seleccionado.		Principal Balance M Gestión Depósito Buscar

Nombre del CU: **Gestionar Sector.**
Actor: Administrador del Balance

Descripción General: El caso de uso se inicia cuando el Administrador del Balance accede al menú Gestión y selecciona la opción Sectores, el sistema muestra las acciones a realizar y, en dependencia de las mismas, muestra el formulario donde debe introducir los datos necesarios para llevar a cabo la acción deseada. El CU termina una vez realizada la operación seleccionada y el Administrador del Balance sale de la aplicación.

Condiciones:
Precondiciones: El usuario debe estar autenticado y poseer los permisos de administración.
Poscondiciones: El sistema queda actualizado luego de realizada cualquier acción descrita anteriormente.

Tabla 8 Matriz Parcial de Escenarios del CU Gestionar Sector

Nombre de la Sección	ID Escenario	Nombre Escenario	Descripción de la Funcionalidad
SC 1: Seleccionar Opción	EC1	Mostrar pestañas	Se muestran dos pestañas “Buscar” y “Adicionar” para la adición, modificación, eliminación y búsqueda de los Sectores.
SC 2: Adicionar Sector	EC2	Adición Exitosa	El usuario selecciona la opción Adicionar Sector, el sistema muestra un formulario que permite insertar un nuevo sector con los campos: nombre del depósito (componente desplegable que permite seleccionar el nombre de los depósitos que han sido adicionados) y nombre del sector, el actor inserta todos los datos y da clic en el botón Enviar. El sistema valida que no exista un sector con el mismo nombre, adiciona el nuevo sector y muestra un mensaje señalando que la adición fue exitosa.
	EC3	Campos Vacíos.	El usuario selecciona la opción Adicionar Sector, el sistema muestra un formulario que permite insertar un nuevo sector con los campos: nombre del depósito (componente desplegable que permite seleccionar el nombre de los depósitos que han sido adicionados) y nombre del sector, el actor deja alguno de los campos en blanco y da clic en el botón Enviar. El sistema marca en rojo los campos en blanco y muestra un mensaje indicando que existen campos vacíos.
	EC4	Sector Existente	El usuario selecciona la opción Adicionar Sector, el sistema muestra un formulario que permite insertar un nuevo sector con los campos: nombre del depósito (componente desplegable que permite seleccionar el

			nombre de los depósitos que han sido adicionados) y nombre del sector, el actor llena los datos pero en el campo nombre de sector entra un nombre que ya existe y da clic en el botón Enviar. El sistema muestra un mensaje de error indicando que ya existe un sector con ese nombre.
SC 3: Buscar Sector	EC5	Buscar Sector	El usuario selecciona la opción Buscar Sector, luego inserta en el campo Buscar cualquier dato que contengan los campos nombre, clasificación, municipio, provincia. El sistema busca el o los sectores de acuerdo a los datos insertados y los muestra en una tabla de la siguiente manera: nombre del sector, nombre del depósito, clasificación, municipio, provincia, Acciones (Modificar, Eliminar, Ver Detalles) y muestra además la opción de generar un documento con los datos de la lista obtenida.
SC 4: Modificar Sector	EC6	Modificación	El actor a partir del flujo de EC5 selecciona la opción Modificar de alguno de los sectores mostrados. El sistema muestra una página que contiene un formulario para modificar los datos del sector (aparecen los mismos campos que en el Adicionar Sector). El usuario modifica los valores que desee y da clic en el botón Enviar. El sistema guarda los datos y muestra un mensaje indicando que la modificación se realizó con éxito.
SC 5: Eliminar Sector	EC7	Eliminación Aceptada	El actor a partir del flujo de EC5 selecciona la opción Eliminar de alguno de los sectores mostrados. El sistema muestra un mensaje solicitando la confirmación de la eliminación. El usuario da clic en el botón Aceptar y el sistema elimina el sector y muestra

			un mensaje indicando que se procedió con la eliminación exitosamente.
	EC8	Eliminación Cancelada	El actor a partir del flujo de EC5 selecciona la opción Eliminar de alguno de los sectores mostrados. El sistema muestra un mensaje solicitando la confirmación de la eliminación. El usuario da clic en el botón Cancelar y el sistema no elimina el sector.
SC 6: Ver Detalles	EC9	Ver Detalles	El actor a partir del flujo de EC5 selecciona la opción Ver Detalles de alguno de los sectores mostrados. El sistema muestra, en una ventana en forma de tabla, toda la información del sector seleccionado.

Tabla 9 Matriz del CP del CU Gestionar Depósito .SC 1: Seleccionar Opción

Escenario	Respuesta del Sistema	Flujo Central
Mostrar pestañas.	El sistema muestra en una pantalla dos pestañas: “Buscar” y “Adicionar”.	Principal Balance M Gestión Sectores

Tabla 10 Matriz del CP del CU Gestionar Sector .SC 2: Adicionar Sector

Escenarios	Nombre del Sector	Nombre del Depósito	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Adición Exitosa	V	V	N/A	El sistema adiciona el sector y muestra un mensaje de confirmación “Se ha insertado un sector”.		Principal Balance M Gestión Sector Adicionar

Campos Vacíos	Algunos de los campos para la adición de un Sector es dejado en blanco.		N/A	El sistema selecciona en rojo los campos que no pueden quedar vacíos y muestra un mensaje: "Existen campos vacíos"		Principal Balance M Gestión Sector Adicionar
Sector Existente	V	V	N/A	El sistema muestra un mensaje de error porque existe otro sector con el mismo nombre: "Ya existe un sector con ese nombre".		Principal Balance M Gestión Sector Adicionar

Tabla 11 Matriz del CP del CU Gestionar Sector .SC 3: Buscar Sector

Escenarios	Nombre del Sector	Nombre del Depósito	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Listar todos los Sectores	V	V	N/A	El sistema busca el o los sectores de acuerdo a los datos insertados y los muestra en una tabla de la siguiente manera: nombre del sector, nombre del depósito, municipio, provincia, Acciones (Modificar, Eliminar, Ver Detalles) y muestra además la opción de generar un documento con los datos de la lista obtenida.		Principal Balance M Gestión Sector Buscar

Tabla 12 Matriz del CP del CU Gestionar Sector .SC 4: Modificar Sector

Escenarios	Nombre del Sector	Nombre del Depósito	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Modificación	V	V	N/A	El sistema modifica los datos del sector y muestra un mensaje: "Sector actualizado".		Principal Balance M Gestión Sector Buscar Modificar

Tabla 13 Matriz del CP del CU Gestionar Sector .SC 5: Eliminar Sector

Escenarios	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Eliminación Aceptada	N/A	El sistema elimina el Sector y actualiza la lista de estos.		Principal Balance M Gestión Sector Buscar Eliminar
Eliminación Cancelada	N/A	El sistema no elimina el Sector.		Principal Balance M Gestión Sector Buscar Eliminar

Tabla 14 Matriz del CP del CU Gestionar Sector .SC 6: Ver Detalles

Escenario	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
Ver Detalles	El sistema muestra, en una ventana en forma de tabla, toda la información del sector seleccionado.		Principal Balance M Gestión Sectores Buscar

2.5 Diseños de Casos de Pruebas de Caja Blanca

2.5.1. Utilizando Técnicas del Camino Básico

Para los CU críticos del **Módulo Inventario de Minerales Sólidos** se determinó seguir la técnica del **Camino Básico**, la cual le permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un *conjunto básico* de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizarán que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. **(8)**

De forma general, los pasos que se deben seguir para la obtención de los casos de prueba en este método, son los siguientes:

1. Emplear el diseño o el código para elaborar el grafo de flujo.
2. Determinar la complejidad ciclomática del grafo de flujo.
3. Determinar un conjunto básico de caminos linealmente independientes.
4. Preparar los casos de prueba que forzarán la ejecución de cada camino del conjunto básico. **(5)**

1. **Caso de Prueba CB: Gestionar Depósito(Modificar Depósito)**
2. **Código**

```
    public function executeModificar()
    {
1   $id= $this->getRequestParameter('id');
2    if($this->getRequest()->getMethod() == sfRequest::POST)
            {
3       $nombre = $this->getRequestParameter('txtNombre');
3       $descri = $this->getRequestParameter('cbxDescripcion');
3          $muni = $this->getRequestParameter('cbxIdMun');
3       $depo=TdepositominPeer::Modificar($id,$nombre,$descri,$muni);
3       $this->redirect('deposito/deposito');
           }
```

4 }

3. Grafo de Flujo de datos

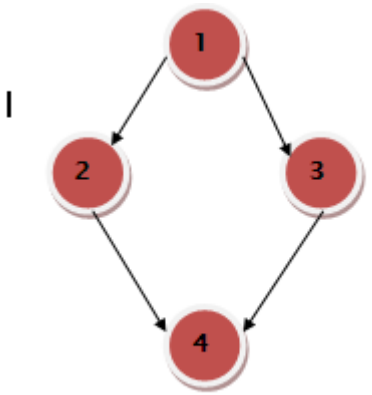


Figura 3: Grafo de flujo de datos para el método Gestionar Depósito (Modificar Depósito)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 4 –4 + 2 = 2

Paso 3: Caminos básicos

CB1: 1 2 4

CB1: 1 3 4

Paso 4: Caso de prueba para el camino básico.

CB1: 1 2 4

Caso de prueba: Gestionar Depósito (Modificar Depósito)

Entrada: nombre, clasificación, provincias y municipios.

Resultado esperado: Se modifican los Datos del Depósito

Resultado de la Prueba: Satisfactoria

1. Caso de Prueba CB: Gestionar Depósito(Adicionar Depósito)
2. Código

```
public function executeAdicionar()
{
1   if($this->getRequest()->getMethod() == sfRequest::POST)
    {
2       $nombre = $this->getRequestParameter('txtNombre');
2       $descri = $this->getRequestParameter('cbxDescripcion');
2       $muni = $this->getRequestParameter('cbxIdMun');
3       if (TdepositominPeer::getDepByNombre($nombre)==null)
        {
4               $depo=TdepositominPeer::adddeposito($nombre,$descri,$muni);
4               $this->getUser()->setFlash('nuevo', $depo);
4               $this->redirect('deposito/deposito ');
        }
        else
        {
5               $this->getUser()->setFlash('mensas', 'Ya existe un deposito con ese
    nombre');
5               $this->redirect('deposito/deposito#adicionar');
        }
    }
6    }
```

3. Grafo de Flujo de Datos:

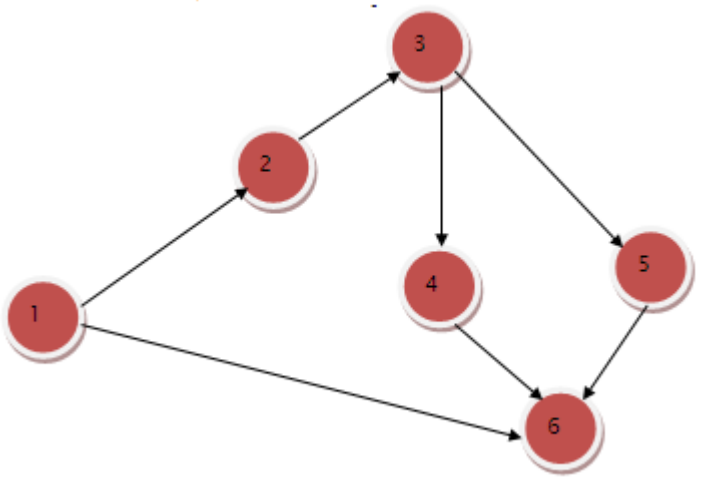


Figura 4: Grafo de flujo de datos para el método Gestionar Depósito (Adicionar Depósito)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 7 –6 + 2 = 3

Paso 3: Caminos básicos

CB1: 1 2 3 5 6

CB2: 1 2 3 4 6

CB3: 1 6

Paso 4: Caso de prueba para el camino básico.

CB1: 1 2 3 5 6

Caso de prueba: Gestionar Depósito (Adicionar Depósito)

Entrada: **nombre, provincia, municipio, clasificación**

Resultado esperado: Se adiciona un nuevo Depósito al Sistema

Resultado de la Prueba: Satisfactoria

1. **Caso de Prueba CB: Gestionar Depósito(Eliminar Depósito)**

2. Código

```
public function executeDelDeposito()
 {
1      $array_json = '{}';
1      $Json = array();
1       try
        {
2      TdepositominPeer::delDepositoId($this->getRequestParameter('id'));
3      $Json[0] = array('estado'=> true, 'men'=>'Dep&oacute;sito Eliminado
    Satisfactoriamente' );
        }
4     catch (Exception $e)
      {
4     $Json[0] = array('estado'=> false, 'men'=> $e->getMessage());
        }
5     $array_json = json_encode($Json);
5    $this->getResponse()->setHttpHeader("application/json");
5    echo $array_json;
5    return sfView::NONE;
6   }
```

3. Grafo de Flujo de datos

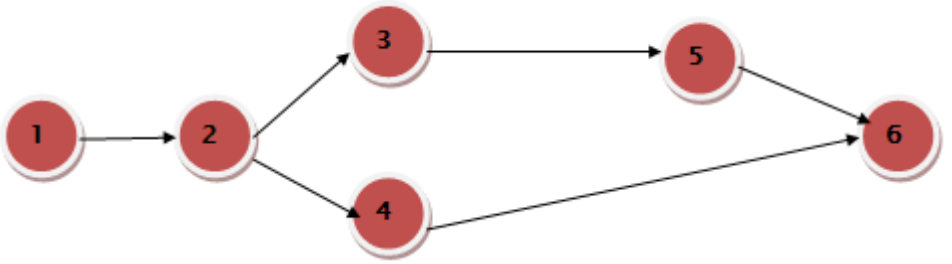


Figura 5: Grafo de flujo de datos para el método Gestionar Depósito (Eliminar Depósito)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 6 –6+ 2 = 2

Paso 3: Caminos básicos

CB1: 1 2 3 5 6

CB2: 1 2 4 6

Paso 4: Caso de prueba para el camino básico.

CB1: 1 2 4 6

Caso de prueba: Gestionar Depósito (Eliminar Depósito)

Entrada**:**

Resultado esperado**: Se elimina un Depósito del Sistema**

Resultado de la Prueba**: Satisfactoria**

1. **Caso de Prueba CB: Gestionar Sector (Adicionar Sector)**

2. **Código**

```
public function executeAdicionar()
    {
    // Propel::getConnection()->executeQuery('set search_path to balancem');
1   $iddep = $this->getRequestParameter('cbxDeposito');
1   $nombre = $this->getRequestParameter('txtNombre');
1       $sect=TsectorminPeer::getSectorerIdDepositoNombresect($iddep,$nombre);
2   if($sect==NUll)
        {
        //Propel::getConnection()->executeQuery('set search_path to balancem');

3           $sector = TsectorminPeer::addsector($iddep,$nombre);
3       $this->getUser()->setFlash('nuevo', $sector);
3         return $this->redirect('sector/sector');
        }
        else {
4                   $this->getUser()->setFlash('mensas', 'Ya existe un sector con ese
  nombre en el
                    deposito');
4                   $this->redirect('sector/sector#adicionar');
                }
5   }
```

3. **Grado de flujo de datos**

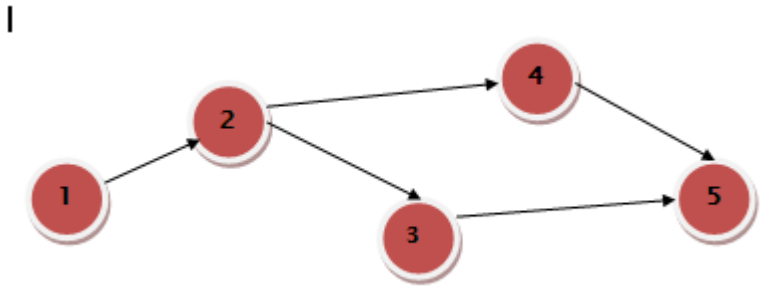


Figura 6: Grafo de flujo de datos para el método Gestionar Sector (Adicionar Sector)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 5 – 5 + 2 = 2

Paso 3: Caminos básicos

CB1: 1 2 4 5

CB2: 1 2 3 5

Paso 4: Caso de prueba para el camino básico.

CB1: 1 2 4 5

Caso de prueba: Gestionar Sector (Adicionar Sector)

Entrada: nombre del Depósito, nombre del Sector

Resultado esperado: Adicionar un Sector

Resultado de la Prueba: Satisfactoria

1. **Caso de Prueba CB: Gestionar Sector (Buscar Sector)**
2. **Código**

```
public function executeBuscarSectoresPage()
 {
1       $texto = $this->getRequestParameter('txtTexto');
1       $inicio = $this->getRequestParameter('txtInicio');
```

```
1   $cantidad = $this->getRequestParameter('txtCantidad');
1   $a= TsectorminPeer::obtenerSectoresTexto($texto, $cantidad, $inicio);
1   $t= TsectorminPeer::CantidadSectoresTexto($texto);
1   $array_json = '{}';
1   $Json = array();
1   $i = 0;
2   foreach ($a as $elem)
        {
3       $dep = TdepositominPeer::getDeposito($elem["id_deposito"]);
3       $nombreDep = $dep->getNombre();
3       //$clasificacion = TelementoPeer::ObtenerNombreElemento($elem["id_clasif"]);
3          $municipio = TelementoPeer::ObtenerNombreElemento($dep-
    >getIdMunicipio());
3       $prov = TprovinciaPeer::getPorIdMunicipio($dep->getIdMunicipio());
3       $provincia = TelementoPeer::ObtenerNombreElemento($prov->getIdProvincia());
3       $Json[$i++] = array (
3       //"Inventarioa" => $elem["id_deposito"],
3       "id" => $elem["id_sector"],
3       "Nombre Sector" => $elem["nombre"],
3       "Municipio" => $municipio,
3       "Nombre Deposito" => $nombreDep,
3       "Provincia" => $provincia
        ) ;
          }
4   $j = 0;
4     $titulos =  array(0 => "Nombre Sector",
4           1 => "Nombre Deposito",
4           2 => "Provincia",
4            3 => "Municipio",
4            4 => "Acciones" ,
```

```
4                5 => "1",
4               6 => "2",
4                7 => "3");
4       $acciones= array(
4                array("accion" => "../sector/modificar/id/",
4                    "img" =>  '../../../../images/icon/edit.png"'),
4                array("accion" => "../sector/delSector/id/",
4                     "img" => '<img src="../../../../images/icon/delete.png"',
4                     "confirmar" => "1",
4                     "mensaje" => "Est&aacute; seguro que desea eliminar el sector",
4                       "titulo" => "Eliminar Sector" ),
4                array("accion" => "../sector/addUni/id/",
4                     "img" => '../../../../images/icon/add.png"'),
                 );
4       $Json[$i++]= $acciones;
4       $Json[$i++] = $titulos;
4       $Json[$i++] = array("total" => $t);
4       $array_json = json_encode($Json);
4       $this->getResponse()->setHttpHeader("application/json");
4       echo $array_json;
4       return sfView::NONE;
5   }

}
```

3. **Grafo de Flujo de datos**

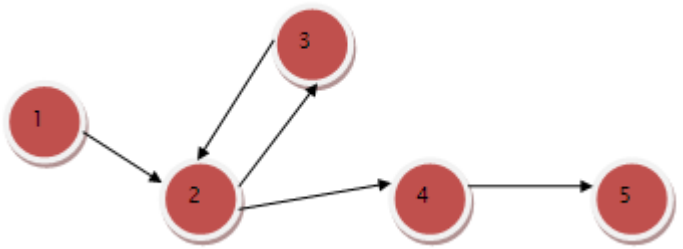


Figura 7: Grafo de flujo de datos para el método Gestionar Sector (Buscar Sector)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 5 – 5 + 2 = 2

Paso 3: Caminos básicos

CB1: 1 2 4 5

CB2: 1 2 3 2 4 5

Paso 4: Caso de prueba para el camino básico.

CB1: 1 2 4 5

Caso de prueba: Gestionar Sector (Buscar Sector)

Entrada: Cualquier dato relacionado tanto con el Depósito o con el Sector (nombre del Sector, provincia, municipio, nombre del Depósito)

Resultado esperado: Buscar un Sector

Resultado de la Prueba: Satisfactoria

1. **Caso de Prueba CB: Gestionar Sector (Modificar Sector)**
2. **Código**

```
public function executeModificarSector()
        {
1     $id=$this->getRequestParameter('id');
1     $nombre = $this->getRequestParameter('txtNombre');
1     TsectorminPeer::Modificar($id,$nombre);
1     return $this->redirect('sector/sector');
    }
```

1. **Grafo de Flujo de Datos**

Figura 8: Grafo de flujo de datos para el método Gestionar Sector (Modificar Sector)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 1

Paso 3: Caminos básicos

CB1: 1

Paso 4: Caso de prueba para el camino básico.

CB1: 1

Caso de prueba: Gestionar Sector (Modificar Sector)

Entrada:

Resultado esperado: Se Modifica el Sector seleccionado para esta acción

Resultado de la Prueba: Satisfactoria

1. **Caso de Prueba CB: Gestionar Sector (Eliminar Sector)**
2. **Código**

```
public function executeDelSector()
   {
1  $array_json = '{}';
1  $Json = array();
1   try
    {
2         $sector = TsectorminPeer::getSector($this->getRequestParameter('id'));
3   $sector->delete();
4         $Json[0] = array('estado'=> true, 'men'=>'Sector Eliminado Satisfactoriamente' );
    }
5   catch (Exception $e)
         {
5        $Json[0] = array('estado'=> false, 'men'=> $e->getMessage());
         };
         $array_json = json_encode($Json);
6        $this->getResponse()->setHttpHeader("application/json");
6        echo $array_json;
6        return sfView::NONE;
7        }
```

3. Grafo de flujo de Datos

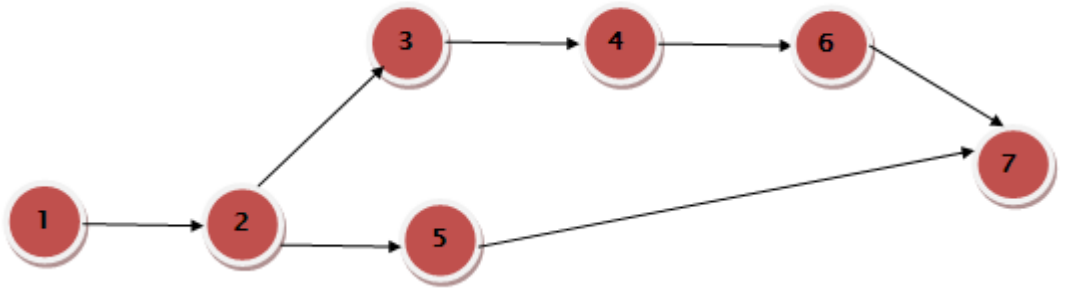


Figura 9: Grafo de flujo de datos para el método Gestionar Sector (Eliminar Sector)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 7 – 7 + 2 = 2

Paso 3: Caminos básicos

CB1: 1 2 5 7

CB2: 1 2 3 4 6 7

Paso 4: Caso de prueba para el camino básico.

CB1: 1 2 5 7

Caso de prueba: Gestionar Sector (Eliminar Sector)

Entrada:

Resultado esperado: Se elimina un Sector Sistema

Resultado de la Prueba: Satisfactoria

2.6 Conclusiones

En este capítulo se describieron de forma específica las características del **Módulo Inventario de Minerales Sólidos** y se realizaron los artefactos necesarios que permitieron lograr un adecuado proceso de pruebas. La realización de la estrategia de pruebas permitió definir como realizar las pruebas y las técnicas a aplicar proporcionando así un conjunto de actividades y técnicas de diseño de casos de pruebas en una serie de pasos bien planificados que dan como resultado la obtención de un producto final con calidad.

CAPÍTULO 3: RESULTADOS

3.1. Introducción

En este capítulo se realiza un análisis de los errores encontrados al aplicar los diseños de caso de prueba desarrollados para las funcionalidades del módulo, para que estos puedan ser corregidos antes de que se entregue el producto, como resultado de lo analizado en los capítulos anteriores, que fueron la base para la realización de los mismos, se hará un resumen de la evaluación de las pruebas, además de hacer un análisis sobre los errores encontrados.

3.2 Resultados de la aplicación de las pruebas al módulo

En este epígrafe se presentan los resultados de la aplicación de las pruebas de caja negra a los casos de uso "Gestionar Depósito" y "Gestionar Sector" según la técnica Partición Equivalente.

3.2.1 Caso de Prueba Gestionar Depósito

Tabla 15 Resultados del CP Gestionar Depósito. SC 1: Mostrar Pestañas

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Mostrar pestañas.	El sistema muestra en una pantalla dos pestañas: "Buscar" y "Adicionar".	Satisfactoria	Principal Balance M Gestión Depósito

Tabla 16 Resultados del CP Gestionar Depósito. SC 2: Adicionar Depósito

Escenario	Nombre del Depósito	Clasificación del Depósito	Provincia	Municipio	Campo Buscar	Respuesta del Sistema	Resultado de la prueba	Flujo Central

Adición Exitosa	Bradier	Yacimiento	Ciego De Ávila(CAV)	Morón	N/A	El sistema adiciona el depósito y muestra un mensaje de confirmación “Se ha insertado un depósito”.	Satisfactoria	Principal Balance M Gestión Depósitos Adicionar
Campos Vacíos	Algunos de los campos se dejan en blanco o se introducen datos de forma incorrecta.				N/A	El sistema selecciona en rojo los campos que no pueden quedar vacíos y no muestra el mensaje “Existen campos vacíos”	Satisfactoria	Principal Balance M Gestión Depósitos Adicionar
Depósito Existente	Bradier	Yacimiento	Pinar del Río (PR)	Pinar del Río	N/A	El sistema muestra un mensaje de error porque existe otro sector con el mismo nombre.	Satisfactoria	Principal Balance M Gestión Depósitos Adicionar

Tabla 17 Resultados del CP Gestionar Depósito.SC 3: Buscar Depósito

Escenario	Nombre del Depósito	Clasificación del Depósito	Provincia	Municipio	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central

Buscar Depósito	Bradier	N/A	N/A	N/A	V	El sistema busca el o los depósitos de acuerdo a los datos insertados y los muestra en una tabla de la siguiente manera: nombre del depósito, municipio, provincia, Acciones (Modificar, Eliminar, Ver Detalles) y muestra además la opción de generar un documento con los datos de la lista obtenida.	Satisfactoria	Principal Balance M Gestión Depósitos Buscar

Tabla 18 Resultados del CP Gestionar Depósito.SC 4: Modificar Depósito

Escenario	Nombre del Depósito	Clasificación del Depósito	Provincia	Municipio	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Modificación	Jesse	Se selecciona la clasificación que se le asignará al depósito	Ciudad de la Habana (CH)	Plaza de La Revolución	V	El sistema modifica los datos del depósito y no muestra el mensaje: “Depósito actualizado”.	Satisfactoria	Principal Balance M Gestión Depósitos Buscar Modificar

Tabla 19 Resultados del CP Gestionar Depósito.SC 5: Eliminar Depósito

Escenarios	Campo Buscar	Respuesta del Sistema	Resultado de la prueba	Flujo Central
Eliminación Aceptada	N/A	El sistema muestra un mensaje “Esta seguro que desea eliminar el Depósito” Se acepta El sistema elimina el Depósito y actualiza la lista de estos.	Satisfactoria	Principal Balance M Gestión Depósitos Buscar Eliminar
Eliminación Cancelada	N/A	Se cancela el mensaje de confirmación El sistema no elimina el Depósito.	Satisfactoria	Principal Balance M Gestión Depósitos Buscar Eliminar

Tabla 20 Resultados del CP Gestionar Depósito.SC 6: Ver Detalles

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Ver Detalles	El actor a partir del EC5, selecciona ver detalles y el sistema muestra, en una ventana en forma de tabla, toda la información del depósito seleccionado.	Satisfactoria	Principal Balance M Gestión Depósito Buscar

3.2.2 Caso de Prueba Gestionar Sector

Tabla 21 Resultados del CP Gestionar Sector.SC 1: Mostrar Pestañas

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central

Mostrar pestañas.	El sistema muestra en una pantalla dos pestañas: “Buscar” y “Adicionar”.	Satisfactoria	Principal Balance M Gestión Sector

Tabla 22 Resultados del CP Gestionar Sector.SC 2: Adicionar Sector

Escenarios	Nombre del Sector	Nombre del Depósito	Campo Buscar	Respuesta del Sistema	Resultado de la prueba	Flujo Central
Adición Exitosa	Praga	Bradier	N/A	El sistema adiciona el sector y muestra un mensaje de confirmación “Se ha insertado un sector”.	Satisfactoria	Principal Balance M Gestión Sector Adicionar
Campos Vacíos	Algunos de los campos para la adición de un Sector es dejado en blanco.		N/A	El sistema selecciona en rojo los campos que no pueden quedar vacíos y no muestra el mensaje: “Existen campos vacíos”	Satisfactoria	Principal Balance M Gestión Sector Adicionar
Sector Existente	Praga	SOS	N/A	El sistema muestra un mensaje de error porque existe otro sector con el mismo nombre: “Ya existe un sector con ese nombre”.	Satisfactoria	Principal Balance M Gestión Sector Adicionar

Tabla 23 Resultados del CP Gestionar Sector.SC 3: Buscar Sector

Escenarios	Nombre del Sector	Nombre del	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central

		Depósito				
Listar todos los Sectores	Cero	Zona2	N/A	El sistema busca el o los sectores de acuerdo a los datos insertados y los muestra en una tabla de la siguiente manera: nombre del sector, nombre del depósito, municipio, provincia, Acciones (Modificar, Eliminar, Ver Detalles) y muestra además la opción de generar un documento con los datos de la lista obtenida.	Satisfactoria	Principal Balance M Gestión Sector Buscar

Tabla 24 Resultados del CP Gestionar Sector.SC 4: Modificar Sector

Escenarios	Nombre del Sector	Nombre del Depósito	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Modificación	Troop23	V	N/A	El sistema modifica los datos del sector y no muestra el mensaje: “Sector actualizado”.	Satisfactoria	Principal Balance M Gestión Sector Buscar Modificar

Tabla 25 Resultados del CP Gestionar Sector.SC 5: Eliminar Sector

Escenarios	Campo Buscar	Respuesta del Sistema	Resultado de la Prueba	Flujo Central

Eliminación Aceptada	N/A	El sistema muestra un mensaje “Esta seguro que desea eliminar el Sector” Se acepta El sistema elimina el Sector y actualiza la lista de estos.	Satisfactoria	Principal Balance M Gestión Sector Buscar Eliminar
Eliminación Cancelada	N/A	Se cancela el mensaje de confirmación El sistema no elimina el Sector.	Satisfactoria	Principal Balance M Gestión Sector Buscar Eliminar

Tabla 26 Resultados del CP Gestionar Sector.SC 6: Ver Detalles

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Ver Detalles	El actor a partir del EC5, selecciona ver detalles y el sistema muestra, en una ventana en forma de tabla, toda la información del depósito seleccionado.	Satisfactoria	Principal Balance M Gestión Sector Buscar

Errores encontrados en el proceso de pruebas

El módulo Inventario de Minerales Sólidos cuenta con 17 casos de uso de los cuales se probaron 6, que eran los que se encontraban implementados en su totalidad cuando se dio inicio al proceso de pruebas. Al finalizar el proceso de pruebas se detectaron un total de 19 no conformidades en el módulo. Dentro de los errores encontrados están los siguientes:

- Existen errores en la validación de los datos. En los campos de entrada de datos, se puede introducir cualquier cadena de texto, no importa cuál sea, la aplicación las acepta.
- Ausencia de mensajes al usuario cuando se dejan campos vacíos en la introducción de datos.

- Cuando el administrador solicita modificar datos y realiza esta acción el sistema no muestra un mensaje de confirmación dónde afirme que los datos han sido modificados.

- El Sistema permite asignar valores impropios, especificamente en el caso de uso Asignar Parámetros de Calidad.

- El Sistema colapsa cuando se introduce el nombre de una unidad existente.

- No existe un soporte para usuarios novatos.

3.3 Validación de especialistas. Método Delphi.

Entre los objetivos de la investigación, se encuentra realizar una validación externa del resultado alcanzado, Para llevar a cabo esta tarea, se hace necesario que cada experto revisando el trabajo previamente hecho, responda la siguiente encuesta:

1. ¿Considera usted que las técnicas de prueba aplicadas al módulo Inventario de Minerales Sólidos fueron las correctas?

2. ¿Considera usted que el proceso de pruebas realizado garantizará que el módulo se entregue libre de defectos?

3. En un rango de 2 a 5 valore la calidad de la investigación.

4. En un rango de 2 a 5 valore la facilidad de comprensión de la investigación.

5. ¿Considera que la investigación se pudiera utilizar como base para la realización del proceso de pruebas en los proyectos existentes en la Facultad?

Tabla 27 Resultados de la encuesta realizada

ENCUESTA	Expertos				
# Pregunta	**1**	**2**	**3**	**4**	**5**
1	**Sí**	**Si**	**No**	**Sí**	**Si**
2	**Si**	**Si**	**Si**	**Si**	**Si**
3	**5**	**3**	**4**	**5**	**5**
4	**5**	**4**	**4**	**5**	**4**
5	**Si**	**Si**	**Si**	**Si**	**Si**

Como se evidencia en la tabla anterior, los expertos en sentido general consideran que la presente investigación se realizó con una calidad aceptable, además piensan que las técnicas y los métodos utilizados para la realización del proceso de pruebas al módulo fueron las correctas. Dos expertos consideran que se deberían de haber realizado también pruebas de carga al módulo, así como la de estrés. Plantean que estas son necesarias debido a la cantidad de peticiones a las que tendrá que responder al mismo tiempo el sistema, otros dos expertos afirman que la investigación si puede ser utilizada como documento de consulta para ciertos aspectos en el proceso de pruebas.

3.4 Conclusiones

Se desarrollaron las pruebas pertinentes en el ámbito adecuado y al analizar los resultados de dichas pruebas realizadas, se puede concluir que la Interfaz de Usuario permite el trabajo cómodo y operativo de los usuarios que utilizan la aplicación. Se realizaron varias recomendaciones sobre algunas particularidades, principalmente los mensajes de orientación al usuario, para lograr el óptimo funcionamiento del sistema. Se informaron adecuadamente todos los errores encontrados para su posterior solución por el equipo de desarrollo del módulo Inventario de Minerales Sólidos.

CONCLUSIONES GENERALES

El aseguramiento de la calidad del software permite reducir notablemente los costos de producción e implantación, y proporciona mayor confianza en el cumplimiento de los requisitos del cliente, siendo las pruebas de software un elemento imprescindible para asegurar la calidad y para verificar el cumplimiento de los requisitos funcionales. Dada la necesidad de que los productos creados sean confiables y precisos, crece la exigencia por parte de clientes y usuarios en general, de que a los sistemas se les hayan aplicado las técnicas de ingeniería de software adecuadas. Al aplicarse las pruebas al módulo Inventario de Minerales Sólidos, se cumplieron todos los objetivos planteados para esta investigación, llegándose a las siguientes conclusiones:

- Con la confección de la Estrategia de Pruebas se logró probar el software a partir de la técnica y enfoque definido.
- Con el diseño de los diversos Casos de Pruebas se logró determinar la cantidad de requisitos funcionales que son parciales o completamente satisfactorios.
- Se detectaron errores significativos, que no fueron descubiertos por los programadores del módulo, quedando probado el cumplimiento de los requisitos funcionales y el adecuado desempeño de los mismos, mediante la aplicación de casos de uso de prueba de caja negra al módulo Inventario de Minerales Sólidos.
- La aplicación de la pruebas de caja negra al módulo Inventario de Minerales Sólidos, permitió ultimar cómo los errores encontrados representan un por ciento poco considerable, pero significativo, siendo este aspecto importante para poder obtener un producto fiable, alcanzando cumplir con las expectativas de los clientes.

RECOMENDACIONES

Luego de haber finalizado el proceso de pruebas al módulo aplicándole las pruebas de caja negra con la técnica partición equivalente, y las pruebas de caja blanca con la técnica camino básico y de haber comprobado que las mismas han arrojado a la luz errores en el sistema que no habían sido detectados en un inicio, se recomienda:

- Realizar pruebas de rendimiento, de carga y de estrés.

REFERENCIAS BIBLIOGRÁFICAS

REFERENCIAS BIBLIOGRÁFICAS

1. **Pérez Martinto, MSc. Pedro Carlos.** *El diseño metodológico de la investigación científica.Teoría de Muestreo: población y muestra. Diseño experimental y métodos. Universidad de las Ciencias Informáticas.*

2. **Pressman, R.** *(2002). Ingeniería del Software: Un enfoque Práctico. McGraw Hill.*

3. **Usaola, Dr. Macario Polo.** Mantenimiento Avanzado de Sistemas de Información Pruebas del Software. http://alarcos.inf-cr.uclm.es/doc/masi/doc/lec/parte5/polo-apuntesp5.pdf . [En línea]

4. **Mario G**. **Piattini**, **Jose A. Calvo-Manzano**, **Joaquin Cervera Bravo**, **and Luis Fernandez Sanz**. *Análisis y diseño de aplicaciones informáticas de gestión, una perspectiva de ingeniería del software*, pages 419-469. Alfaomega, 2004.

5. **Pressman, Roger S.** Ingeniería del software, un enfoque práctico, pages 281-322. McGrawHill, quinta edition, 2002. [En línea]

6. **Myers, Glenford J.** *The art of sotware Testing, page 234.Wiley, second edition, 2004.*

7. **Patton, Ron.** *Software Testing, page 408.Sams Publishing, 2005.*

8. **MacCbe, T.** *A software complexity measure.IEEE Trans. Software Engineering, 2:308-320, 1976.*

9. **Tai, K.C.** What to do beyond branch testing.ACM Software Engineering Notes, 14:58-61,1989.

10. *Ingeniería del Software, Un Enfoque Práctico. Holguín : Empresa Poligráfica José Miró Argenter, 2005. págs. 312-314. Vol. 1.*

11. *Ingeniería del Software, Un Enfoque Práctico. Holguín : Empresa Poligráfica José Miró Argenter, 2005. págs. 316-318. Vol. 1.*

12. **Javier J. Gutiérrez, María J. Escalona, Manuel Mejías y Antonia M. Reina.** MODELOS PARA PRUEBAS DEL SISTEMA.

13. **Cockburn, A.** *2000. Writing Effective Use Cases. Addison-Wesley 1st edition. USA.*

14. **E, Dustin.** *et-al. 2002. Quality Web Systems. Performance, Security, and Usability.*

15. **M.J., Escalona.** *2004. Models and Techniques for the Specification and Analysis of Navigation in Software Systems. Ph. European Thesis. Department of Computer Language and Systems. University of Seville. Seville, Spain.*

16. *El Proceso Unificado de Desarrollo de Software. La Habana : Félix Varela, 2004. Vol. 1.*

17. **Pressman, Roger S.** *Ingeniería del Software, Un Enfoque Práctico.* Holguín: Empresa Poligráfica José Miró Argenter, 2005. págs. 294-298. Vol. 1.

18. **Myers, Glenford J.** *The Art of Software Testing.* New York : s.n., 1979.

19. **Boris Beizer.** *Software Testing Techniques*. Van Nostrand Reinhold, second edition, 1990.

20. Metodología MÉTRICA versión 3. *Técnicas y Prácticas*. Ministerio de Administraciones Públicas, 2002.

21. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James**. *El Proceso Unificado de Desarrollo de Software.* La Habana: Félix Varela, 2004. págs. 1-12. Vol. 1.

22. **Jacobson**, **B**., **Rumbaugh**. , El Proceso Unificado de desarrollo de Software, 2004. Volumen 1,Editorial Félix Varela: p. Capítulo 11 pag 281-304.

23. Mario G. Piattini, Jose A. Calvo-Manzano, Joaquín Cervera Bravo, and Luis Fernández Sanz. *Análisis y diseño de aplicaciones informáticas de gestión, una perspectiva de ingeniería del software*, pages 419-469. Alfaomega, 2004.

24. Metodología Métrica versión 3. *Técnicas y Prácticas. Ministerio de Administraciones Publicas 2002.*

25. **A. Davis**.*201Principles of Software Development*. **McGraw-Hill**, 1995.

BIBLIOGRAFÍA

BIBLIOGRAFÍAS CONSULTADAS

1. García, Lourdes, Álvarez, Sofía. Una visión general sobre la certificación de calidad y la evaluación del proceso de desarrollo de software. Rev. Ingeniería Industrial. No.

2. R. S. Pressman. *Ingeniería del software. Un enfoque práctico*. 4ªEdición. McGrawHill (1998)
IEEE *Standard Glossary of Software Engineering Terminology*, 1990.

3. Collazo, Manuel. *Técnicas de prueba del software. Estrategias de prueba del software.* 2003.

4. Rodríguez, D. E. *Introducción a la Calidad del Software*, Septiembre, 2003.

5. NOVATICA. Número 137, Enero-Febrero 1999. Monográfico Calidad del Software / Software de calidad.

6. Pérez Martinto, MSc. Pedro Carlos. El diseño metodológico de la investigación científica.Teoría de Muestreo: población y muestra. Diseño experimental y métodos. Universidad de las Ciencias Informáticas.

[En línea]

7. Mañas, J. A. *Pruebas de Programas*, 28 de Febrero de 2002.

8. M. Vásquez C. Falcato Editorial Prensa Técnica S. L. España

9. Anna C. Grimán, M. P. (2005). Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales. Caracas, Venezuela: Departamento de Procesos y Sistemas. Universidad Simón Bolívar.

10. Cosín, J. D. (2007). ***Técnicas cuantitativas para la gestión en la ingeniería del software***

11. L. S., & Y. P. (2007). *Diseño y aplicación de pruebas al producto Registro Cubano de Discapacitados. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* Ciudad de la Habana. Cuba.

12. Sommerville, I. (2005). *Ingeniería del software. Séptima Edición.* Madrid: Pearson Educación S.A.

13. Díaz, J. G. *Introducción al Proceso de Pruebas.* Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla., Sevilla.

14. Usaola, Dr. Macario Polo. Mantenimiento Avanzado de Sistemas de Información Pruebas del Software. http://alarcos.inf-cr.uclm.es/doc/masi/doc/lec/parte5/polo-apuntesp5.pdf . [En línea]

15. Villalobos Hernández María de la luz, A. F. G. *Investigación sobre las prácticas de ingeniería de software en México*, Agosto, 2001.

16. Ramírez, Jaime. Métodos de Prueba del Software. Unidad de Programación,2007.

17. CARVAJAL, L. G. A.-P. H. Eficiencia de la caracterización de técnicas de prueba en un contexto real de aplicación.

18. A.Davis.201 Principles of Software Development. McGraw-Hill, 1995.

19. Roger S. Pressman. Ingeniería del software, un enfoque práctico, páginas 281-322. McGrawHill, quinta edición, 2002.

20. Javier J. Gutiérrez, María J. Escalona, Manuel Mejías y Antonia M. Reina. MODELOS PARA PRUEBAS DEL SISTEMA.

21.*Ingeniería del Software, Un Enfoque Práctico. Holguín : Empresa Poligráfica José Miró Argenter, 2005. págs. 312-314. Vol. 1.*

22. *Ingeniería del Software, Un Enfoque Práctico. Holguín : Empresa Poligráfica José Miró Argenter, 2005. págs. 316-318. Vol. 1.*

23. Pérez Martinto, MSc. Pedro Carlos. El diseño metodológico de la investigación científica.Teoría de Muestreo: población y muestra. Diseño experimental y métodos. Universidad de las Ciencias Informáticas.

24.Mario G. Piattini, Jose A. Calvo-Manzano, Joaquin Cervera Bravo, and Luis Fernandez Sanz. Análisis y diseño de aplicaciones informáticas de gestión, una perspectiva de ingeniería del software, pages 419-469. Alfaomega, 2004.

25. Handbook of walkthroughs, Inspections and Technical Reviews, 3ra edic**ión Freeman, D.P., Weimberg, G.M., Dorset House,1990**

26. "Implementating Software Inspections", Notas del curso, IBM Systems Sciences Institute, IBM Corporation, 1981

27. Jones, T.C., Programming Productivity, McGraw-Hill, 1986

28. Software Inspection. An industry Best Practice, Wheeler, DA, Brykezyski, B, Meeson, RN, IEEE Computer Society Press, 1996

29. Fowler, Martin; Kendall Sccott (1999). *UML Gota a Gota* (en Español), Addison Wesley

30. Haeberer, A. M.; P. A. S. Veloso, G. Baum (1988). *Formalización del proceso de desarrollo de software*, Ed. preliminar edición (en Español), Buenos Aires: Kapelusz.

31. JACOBSON; BOOCH; RUMBAUGH (1999). ***UML -*** *El Lenguaje Unificado de Modelado* (en Español), Pearson Addisson-Wesley. Rational Software Corporation, Addison Wesley Iberoamericana

32. JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James (2000). *El Proceso Unificado de Desarrollo de Software* (en Español), Pearson Addisson-Wesley.

33. Pressman, Roger S. (2003). *Ingeniería del Software, un enfoque Práctico*, Quinta edición edición (en Español), Mc Graw Hill.

GLOSARIO DE TÉRMINOS

Glosario de Términos

Calidad: Calidad de software: Satisfacción de las necesidades de los usuarios.

Caso de prueba: Especificación de un caso para probar el sistema, incluyendo qué probar, con qué entradas y resultados y bajo qué condiciones.

Prueba: Ejecución de un programa para verificar si cumple con lo establecido.

Estrategias de Pruebas: Describe 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

Niveles de Pruebas: Pruebas que se aplican en diferentes escenarios y en las diferentes etapas del proceso de desarrollo.

Resultados de Pruebas: Permiten hacer una evaluación precisa del software.

ANEXOS

Lista de Chequeo.

En el documento Manual de usuario:

¿Se especifica detalladamente cada interfaz del flujo de eventos a probar?

¿Se explicitan todos los campos y funcionalidades de éstos por interfaz?

¿Corresponde la verdadera interfaz de usuario con la señalada en el documento?

¿Se han identificado errores ortográficos?

¿Se entiende claramente lo que se ha especificado en el documento?

En el documento de Especificación de requisitos:

¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?

¿Debería especificarse algún requisito con más detalle?

¿Todos los requisitos identificados se centran en lo que el sistema debe hacer y no como el sistema debe hacerlo?

¿Se han identificado los requisitos de software y de hardware?

¿Los requisitos de soporte y usabilidad se han identificado?

¿Se puede verificar cada requisito? (Un requisito se dice que es verificable si existe algún proceso no excesivamente costoso por el cual una persona o una máquina pueda chequear que el software satisface dicho requisito, ejemplo la especificación del caso de uso).

¿Se han enumerado los requisitos incluso los que se derivan de otros requisitos?

¿Se han identificado errores ortográficos?

¿Se entiende claramente lo que se ha especificado en el documento?