Universidad de las Ciencias Informáticas

Facultad 9

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

Título: Sistema de Gestión de Datos Geológicos. Rol Diseñador de Casos de Prueba, Módulo Búsqueda Referativa.

Autor: Alexei Rojas Doimeadios
Tutor: Ing. Rubén Darío Jardines Guevara

Ciudad de la Habana. “Año 52 de la Revolución”
jueves, 24 de junio de 2010

Agradecimientos

A mi mamá y mi papá por darme este tamaño y apoyándome en mis decisiones, confiando en mi. Siempre siendo los principales guías en mi desarrollo...siendo las personas fundamentales en mi vida.

A mis abuelos Oscar y Teresa Doimeadios, personas fundamentales en mi niñez y adolescencia, los encargados de guiarme durante casi toda mi vida infantil, por su cariño y dedicación.
A mi hermano Andrei Rojas Doimeadios, por ser un ejemplo a seguir cuando las situaciones son desfavorables, por su amor inigualable de hermano. Por ser siempre un apoyo para mi vida como futuro profesional.

A los chicos que conocí cuando empezaba mis estudios y más tarde consideré amigos, porque aprendí a confiar en ellos y de los que recibí apoyo y cariño: Popi, Reinier, Raúl, Bárbaro, Yasmany y demás amigos que siempre estuvieron ahí y me apoyaron cada vez que los necesite.

A todos mis amigos de la secundaria, del PRE y de la Universidad en general.

A mis vecinos y amigos por quererme tanto y preocuparse por mí.

A todos los profesores que me ayudaron en mi preparación profesional
A mi tutor Rubén Darío Jardines por guiarme durante el desarrollo de la tesis.
A los compañeros del proyecto.

A Natalia Muñoz Bairan mi futura esposa y madre de mis hijos, gracias por estar a mi lado, gracias por aguantar mi pesadez, por abrazarme todas las noches, por tu preocupación en momentos malos que nos pueda traer la vida, tanto a ti como a mí.

Gracias a todos por la comprensión y la dedicación que me han brindado...se les quiere a todos...!!!!

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 2010

Autor: Alexei Rojas Doimeadios

_____

Tutor: Ing. Rubén Darío Jardines Guevara

_____

Resumen

El presente trabajo tiene como objetivo desarrollar las pruebas de software al Módulo Búsqueda Referativa que forma parte del Sistema de Gestión de Datos Geológicos (SGDG) como línea de producción del Centro Geoinformática y Señales Digitales (GEySED) de la facultad 9, perteneciente a la Universidad de las Ciencias Informáticas.

Entre los tópicos que se trataron se encuentra una breve argumentación de los niveles, técnicas y métodos de prueba existentes; dentro de las pruebas que se efectuaron se encuentran: los métodos de prueba de caja negra y caja blanca, utilizando las técnicas de partición equivalente y camino básico respectivamente. Además se efectúan pruebas a la documentación haciendo uso de lista de chequeo, las cuales evalúan y miden la calidad del documento para hacer entrega del mismo libre de errores.

Para la realización de las pruebas se planificó una estrategia a seguir para guiar el proceso y finalmente se realizó la evaluación de los resultados obtenidos.

Se diseñó un plan de prueba, casos de prueba y se documentaron los resultados de la aplicación de los mismos, atendiendo a las características del sistema desarrollado, lo que permitió la detección de los errores de los casos de uso y de la lógica del programa de la aplicación.

Palabras claves: Pruebas, errores, casos de prueba.

Índice de Ilustraciones

Índice de Tablas

Tabla de contenido

Introducción

En los últimos años, el software ha pasado de ser una solución especializada a los problemas y un objeto de análisis, a ser una industria, creciendo de forma apresurada y con ello buscando la mayor satisfacción de los clientes en el menor tiempo y costo posible. Hoy día, la calidad, tanto en la industria como en el comercio y en las organizaciones de servicios y nuevas tecnologías constituye un claro factor diferencial competitivo; dentro de las cuales, la industria del software no es una excepción.

El método intuitivo de desarrollo de software que se usa actualmente es, básicamente, el trabajo en equipos de desarrollo, los cuales inevitablemente serán propensos a cometer errores, es por ello que se hace cada vez más imponente el tema de la calidad del software en todas las esferas donde este se desarrolle.

Las compañías dedicadas al desarrollo de software pretenden aumentar la calidad de sus productos para evitar los problemas inherentes a sus procesos: plazos y presupuestos quebrantados, insatisfacción del usuario, insuficiente productividad y la baja calidad lograda. Por los requerimientos evidenciados anteriormente, es necesario y de suma importancia hacer software de buena calidad para lograr ingresar en el mercado.

En Cuba, se han determinado líneas de acción de interés para el Estado, que repercutirán en la economía, política y la sociedad, buscándose beneficios que traigan consigo no solo un desarrollo de productos de software, sino también su incursión en el mercado a escala mundial aprovechando su perspectiva económica. La coyuntura económica de Cuba favorece en este momento el desarrollo de sistemas, propiciando no solo el uso de estos en una organización o entidad nacional sino también para la comercialización a otras naciones o entidades interesadas.

La Universidad de las Ciencias Informáticas (UCI) desempeña un papel importante, pues su objetivo principal es producir software para Cuba y comerciar sus productos con el resto del mundo. Para el cumplimiento de este objetivo la universidad tiene proyectos productivos con disímiles contenidos y que involucran diferentes áreas del conocimiento; y para lograr un nivel óptimo de calidad en cada producto que se desarrolle, dentro del proceso de aseguramiento de calidad de los mismos, se hace necesario llevar a cabo una estrategia de pruebas que asegure el cumplimiento de los requisitos de calidad.

Uno de los productos con que cuenta la UCI es el Sistema de Gestión de Datos Geológicos (SGDG), que está conformado por 8 módulos de los cuales uno es el Módulo Búsqueda Referativa, que se encarga de conservar y administrar la documentación geólogo-minero-petrolera que se ha generado en el país desde hace más de 100 años. En este momento cuenta con más de 5500 informes relacionados con las actividades geológicas, mineras y petroleras realizadas en el país por un sinnúmero de empresas y compañías especializadas. A su vez, el módulo gestiona igualmente la información referida a los préstamos de estos informes y sus respectivos procesos de devolución. (Fernández, 2009.)

El sistema está desarrollado para la Oficina Nacional de Recursos Minerales (ONRM), entidad destinada a fiscalizar y controlar la actividad minera y el uso racional de los recursos minerales, también asesora al Ministerio de la Industria Básica(MINBAS) y demás organismos de la administración central del estado sobre las actividades mineras de exploración-producción de hidrocarburos, además es el depositario de la información geológica, minera y petrolera de la nación, recibe, organiza y conserva la información y brinda servicios de información técnica. (Fernández, 2009.)

La realización de este sistema ha implicado un gran compromiso por parte de cada uno de los miembros del equipo de desarrollo con el cliente, para conseguir el agrado del mismo en todos los sentidos; aunque se realizó un buen trabajo por parte del equipo de desarrollo la existencia de errores es posible, por lo que se hace necesario encontrar y documentar los defectos en la calidad de los requerimientos del sistema del Módulo Búsqueda Referativa, además después de la construcción del sistema es evidente la necesidad de validar que este módulo funcione como fue diseñado y que los requerimientos fueron implementados apropiadamente. Es por tal motivo que se convierte en una necesidad, el desarrollar un proceso de pruebas para mitigar las situaciones de desavenencias entre los requerimientos del sistema y lo que se automatiza antes de poner en uso el producto por el cliente.

Dada la situación anterior se presenta el siguiente **problema a resolver** *¿Cómo lograr la entrega del Módulo Búsqueda Referativa libre de defectos?*

Para dar solución al problema planteado se define el **objeto de estudio** como *el proceso de pruebas de sistemas de gestión de información*, y como **campo de acción** *el diseño e implementación de las pruebas del Módulo Búsqueda Referativa.*

Del problema planteado se propone que *el diseño y la implementación de las pruebas al Módulo Búsqueda Referativa que forma parte del Sistema de Gestión de Datos Geológicos garantizarán que el mismo se entregue libre de defectos* siendo esta afirmación la **idea a defender**, siendo el **objetivo general** *la documentación técnica del proceso de pruebas correspondiente al Módulo Búsqueda Referativa que forma parte del Sistema de Gestión de Datos Geológicos.*

Para darle cumplimiento al objetivo general se proponen las siguientes **tareas investigativas**:

- Caracterizar el proceso de pruebas según la metodología seleccionada.
- Caracterizar el rol Diseñador de Casos de Prueba según la metodología seleccionada.
- Elaborar el plan de pruebas para el módulo Búsqueda Referativa.
- Diseñar los Casos de Prueba
- Realizar las pruebas de caja blanca al módulo Búsqueda Referativa.
- Realizar las pruebas de caja negra al módulo Búsqueda Referativa.
- Caracterizar el estado de la implementación una vez finalizado el proceso de pruebas.
- Validar las pruebas realizadas.

Con el desarrollo de la investigación se podrán obtener como **posibles resultados:** *La evaluación y validación de la implementación del Módulo Búsqueda Referativa a partir del desarrollo del proceso de pruebas.*

En el transcurso de la investigación, se ponen en práctica métodos científicos como los métodos teóricos y empíricos:

Los métodos teóricos permiten estudiar las características del objeto de investigación que no son observables directamente, facilitan la construcción de modelos e hipótesis de investigación y crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad, contribuyendo al desarrollo de las teorías científicas y para su ejecución se apoyan en el proceso de análisis y síntesis. Posibilitan el conocimiento del estado del arte del fenómeno, su evolución en una etapa determinada, su relación con otros fenómenos, así como su aislamiento como objeto estudiado. (Martinto)

Los métodos teóricos utilizados en la investigación se enumeran a continuación.

- **Método histórico-lógico:** Se utilizó para investigar la información nacional e internacional que se tiene hasta el momento sobre las pruebas y elaborar un estado del arte sobre el tema.

- **Método analítico sintético:** Se utilizó en la división de los aspectos obtenidos sobre el desarrollo de pruebas en partes para un mejor estudio y poder analizar las características de las mismas dentro del flujo de desarrollo del software y resumir los aspectos fundamentales necesarios para la investigación en curso.

- **Método de modelación:** Se utilizó 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, además siempre que se ponga en práctica el desarrollo de gráficas de apoyo o mapas conceptuales.

Los métodos empíricos permiten extraer de los fenómenos analizados las informaciones que se necesitan sobre ellos a través de observaciones, del uso de técnicas opináticas y la propia experimentación. (Martinto)

Los métodos empíricos utilizados en la investigación se enumeran a continuación.

- **Observación:** es utilizado para comprobar que los requerimientos fueron implementados correctamente.

- **Encuesta:** es utilizado para validar los resultados obtenidos.

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

- **Capítulo 1:** “Fundamentación teórica”, se describe todo lo relacionado al objeto de estudio, así como los términos utilizados y el análisis de otras documentaciones existentes para el Rol Diseñador de Casos de Prueba para el Módulo Búsqueda Referativa que forma parte del SGDG.

- **Capítulo 2:** “Diseño y aplicación de las pruebas”, se elabora un plan de prueba que dará solución al problema de forma satisfactoria. Finalmente, se brinda el diseño de casos de prueba y se realizan las pruebas de caja blanca y de caja negra.
- **Capítulo 3:** “Resultados”, se caracteriza el estado de la implementación y se validan las pruebas realizadas en el capítulo anterior al Módulo Búsqueda Referativa del proyecto SGDG.

Capítulo 1 : Fundamentación Teórica.

1.1 Introducción

En este capítulo se abordarán los temas fundamentales referidos a la ingeniería de pruebas, conceptos y definiciones, aspectos relacionados con la calidad, los niveles, métodos y técnicas de diseño de pruebas, procedimientos y plan de prueba, destacando algunos criterios generales, de los cuales se presentará su definición. Además se hará referencia a las definiciones que propone el Proceso Unificado de Rational (RUP) como metodología de desarrollo para este proceso.

1.2 Conceptos asociados al dominio del problema

El concepto de calidad ha evolucionado en el tiempo y en dependencia de la profesión de la persona que la estudie y la utilice como herramienta en la gerencia de las industrias.

La calidad es una palabra clave de la competitividad. Las empresas en el mundo se han dado cuenta que el cuidado de los detalles, lograr la satisfacción del cliente y la confianza que éstos tengan con el producto permitirá permanecer y crecer en el mercado u obligará a salir de él.

1.2.1 ¿Qué es la calidad del software?

En un mundo globalizado, donde las organizaciones se ven enfrentadas a competencias de nivel mundial, la calidad se convierte en un importante punto diferenciador, además de aumentar la satisfacción general del cliente, disminuir costos y optimizar los recursos. Los productos o servicios que ostentan certificados de calidad son preferidos por los compradores porque trasmiten seguridad y confianza. Esto también constituye un atributo de valor para las estrategias de comercialización en el exterior.

Según Pressman la calidad del software se define como: *Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente.* (Pressman, 2002)

La calidad del software es una preocupación a la que se dedican muchos esfuerzos. Todo proyecto tiene como objetivo producir software de la mejor calidad posible, que cumpla y supere las expectativas de los usuarios.

La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia, es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad.

1.2.1.1 Factores que miden la calidad del software

Un aspecto importante de conocimiento del tema de la calidad del software, son los factores que determinan la misma, estos se centran en 3 aspectos importantes de un software:

- **Características operativas:**
  - **Corrección:** hasta dónde satisface un programa su especificación y logra los objetivos del cliente.
  - **Fiabilidad:** hasta dónde se puede esperar que un programa lleve a cabo la función con la que fue concebido, con la exactitud requerida.
  - **Eficiencia:** la cantidad de recursos informáticos y de códigos necesarios para que un programa funcione.
  - **Seguridad:** hasta dónde se puede controlar el acceso al software o a los datos por personas no autorizadas.
  - **Facilidad de uso:** aprender a operar los datos de entrada e interpretar las salidas de un programa.
- **Capacidad de soportar los cambios:**
  - **Facilidad de mantenimiento:** la capacidad para localizar y arreglar un error en un programa.
  - **Flexibilidad:** el esfuerzo necesario para modificar un programa operativo.
  - **Facilidad de prueba:** el esfuerzo necesario para probar un programa para asegurarse de que realiza su función pretendida.
- **Adaptabilidad a nuevos entornos:**
  - **Portabilidad:** la facilidad para transferir el programa de un entorno de sistema hardware y/o software a otro entorno diferente.
  - **Reusabilidad (Capacidad de reutilización):** hasta dónde se puede volver a emplear un programa (o partes de un programa) en otras aplicaciones.
  - **Interoperabilidad:** el esfuerzo para acoplar un sistema con otro. (Rubio, 2002)

1.3 Descripción del objeto de estudio

Un producto antes de ser liberado es necesario evaluar la calidad del mismo, lo principal para este aspecto son los procesos de prueba que permiten verificar y revelar la calidad de un producto software.

El proceso de prueba posibilita que al final se garantice que la calidad del software aumente, al detectarse fallas en el sistema, estas se puedan corregir a tiempo, antes de que el producto se entregue a los clientes.

1.3.1 Metodología a utilizar en la investigación

El desarrollo de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo. A la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software, el objetivo de una metodología es controlar de manera transparente todo el proceso de desarrollo.

La metodología a seguir en el proyecto SGDG según la arquitectura de software definida para su desarrollo es el Proceso Unificado de Rational (RUP). Que define la representación arquitectónica mediante las 4+1 vistas arquitectónicas: la vista de casos de uso, la vista lógica, la vista de procesos, la vista de implementación y la vista de despliegue. En el documento Arquitectura de Software del proyecto no se muestra una vista de procesos debido a que el sistema no presenta procesos concurrentes de gran relevancia y se muestra además una vista de datos. (Ronquillo, 2009)

¿Qué es RUP?

Ilustración 1. Proceso Unificado de Rational (RUP)

RUP brinda una guía para encontrar, organizar, documentar, y seguir los cambios de los requisitos funcionales y restricciones. Utiliza una notación de Caso de Uso y escenarios para representar los requisitos.

RUP es una metodología que se encarga muy bien de ordenar el flujo de trabajo, porque es un proceso que integra las múltiples facetas del desarrollo (tiene 9 flujos de trabajo, de los cuales 6 son ingenieriles y los demás de gestión, a su vez está integrado por 4 fases en la cuales se distribuyen las actividades a realizar, dando como resultado los artefactos de cada uno). (Soto, 2007)

RUP promueve el desarrollo iterativo e incremental y organiza el desarrollo de software y sistemas en cuatro fases.

El proceso puede ser descrito en dos dimensiones o ejes. (Corporation, 1998):

Eje horizontal: Representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso expresado en términos de fases, iteraciones e hitos. Se puede observar en la Ilustración 2 que RUP consta de cuatro fases: Inicio, Elaboración, Construcción y Transición.

Eje vertical: Representa los aspectos estáticos del proceso. Describe el proceso en términos de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

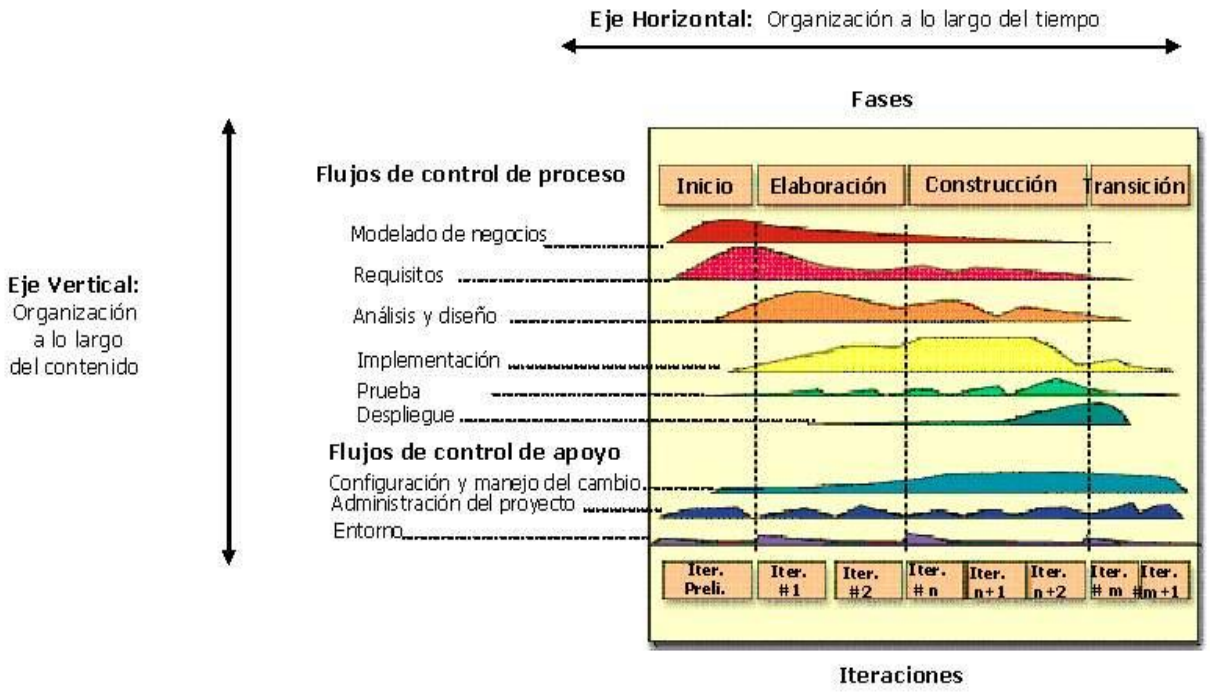

Ilustración 2. Estructura de RUP

Características de RUP:

- Iterativo e Incremental, es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

- Dirigido por los casos de uso, estos proporcionan un hilo conductor ya que avanza a través de una serie de flujos de trabajos que parten de ellos.

- Centrado en la arquitectura, muestra la visión común del sistema completo en la que el equipo de desarrollo y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

1.3.2 Proceso de Pruebas de Software de RUP

A toda organización le gustaría mejorar el modo en que opera tanto si supone aumentar su participación en el mercado, reducir los costos, gestionar los riesgos con mayor eficacia, como mejorar la satisfacción de los clientes.

Las pruebas de software constituyen un pilar indispensable para evaluar y determinar la calidad de un software. Se pueden definir las pruebas de software como:

- El proceso de evaluación de un producto desde un punto de vista crítico, donde el "probador" (persona que realiza las pruebas) somete el producto a una serie de acciones, y el producto responde con su comportamiento como reacción. (Collazo, 2003)
- *«Una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.»* (Collazo, 2003)

Tienen como objetivos:

- Encontrar y documentar los defectos que pueden afectar la calidad del software.
- Verificar que el software trabaje como fue diseñado.
- Validar y probar los requisitos que debe cumplir el software.
- Validar que los requisitos fueron implementados correctamente.

Las pruebas dan como resultado algunos aspectos como son:

- Más importancia y protagonismo día a día.
- Garantizan la calidad del software.
- Garantizan la satisfacción de los requisitos.
- Ahorran tiempo y recurso en el desarrollo.

Las pruebas permiten validar y verificar el software, entendiendo como validación el proceso que determina si el software satisface los requisitos, y verificación como el proceso que determina si los productos de una fase satisfacen las condiciones de la misma.

1.3.3 Trabajadores definidos en RUP para el flujo de trabajo de prueba

Los trabajadores definidos en RUP para el flujo de trabajo de prueba son:

- Administrador de Prueba: Es el responsable del éxito de la prueba, Este rol involucra defensor de prueba y calidad, planificación y administración de recursos, resolución de problemas que impiden las pruebas. (Pressman, 2004)

- Analista de Prueba: es el responsable de identificar y definir las pruebas requeridas, monitorear el progreso de la prueba y el resultado en cada ciclo de prueba y evaluando la calidad total experimentada como un resultado de las actividades de prueba. (Pressman, 2004)

- Diseñador de prueba: planifica las pruebas. Es el responsable de definir el método de prueba y asegurar su implementación exitosa. El rol incluye identificando técnicas apropiadas, herramientas e instrucciones para implementar las pruebas necesarias y encauzar los recursos correspondientes para las pruebas. (Pressman, 2004)

- Probador: es el responsable durante las actividades principales de las pruebas, el cual incluye la conducción de las pruebas necesarias y el registro del resultado de la prueba. (Pressman, 2004)

1.3.4 Artefactos del flujo de trabajo de prueba

El Proceso Unificado de Rational plantea 7 artefactos fundamentales para el flujo de trabajo de prueba los cuales deben ser generados dentro de este flujo.

A continuación se enumeran y describen los artefactos que genera RUP en el proceso de prueba:

✓ **Modelo de prueba**

Describe como se prueban los componentes en el modelo de implementación ejecutando pruebas de integración y de sistema, este artefacto puede describir también como han de ser probados aspectos específicos del sistema, por ejemplo si la interfaz de usuario del sistema cumple con su objetivo y describe el cumplimiento de los requisitos funcionales y no funcionales. Es una colección de casos de prueba, procedimientos de prueba y componentes de prueba. (I. Jacobson, 2000)

✓ **Caso de prueba**

Es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular ó una función esperada. La entidad más simple que siempre es ejecutada como una unidad, desde el comienzo hasta el final. (I. Jacobson, 2000)

✓ **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 monitorizando la ejecución de los componentes a probar. (I. Jacobson, 2000)

✓ **Procedimiento de prueba**

Especifica cómo realizar uno o varios casos de prueba o partes de estos, un procedimiento de prueba puede ser tan simple como por ejemplo una instrucción para un usuario 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 prueba. (I. Jacobson, 2000)

✓ **Plan de Prueba**

Describe la estrategia, recursos y planificación de las pruebas. La estrategia de prueba incluye la definición del tipo de prueba a realizar para cada iteración y sus objetivos, el nivel de cobertura de prueba y el porcentaje de prueba que deberían ejecutarse con un resultado específico. (I. Jacobson, 2000)

✓ **Defecto**

Es un síntoma de un fallo de software o un problema descubierto en una revisión, el cual puede ser utilizado para localizar cualquier cosa que los desarrolladores necesiten registrar cómo síntoma de un problema en el sistema. (I. Jacobson, 2000)

✓ **Evaluación de prueba**

Es una evaluación de los resultados de los esfuerzos de prueba, tales como la cobertura del caso de prueba, la cobertura del código y el estado de los defectos. (I. Jacobson, 2000)

1.3.5 Artefactos generados en el proceso de pruebas al Módulo Búsqueda Referativa

En el proceso de pruebas a realizar al Módulo Búsqueda Referativa quedan definidos los siguientes entregables.

Plan de prueba: Dicho plan está diseñado para asegurar que se satisfacen todos los requisitos funcionales especificados por el usuario teniendo en cuenta también los requisitos no funcionales relacionados con el rendimiento, seguridad de acceso al sistema, a los datos y procesos, así como a los distintos recursos del sistema. Se incluyen las especificaciones de Software y Hardware, teniendo en cuenta las herramientas y los dispositivos que se necesitan para el desarrollo del sistema.

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

Informe de No Conformidades: Documento en el cual se van a archivar los incumplimientos de los requisitos que deben ser cumplidos en el proyecto. Este documento define el procedimiento interno a seguir por los desarrolladores del proyecto para el tratamiento de los errores cometidos.

1.3.6 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.
- Implementar prueba.
- Evaluar pruebas.

1.3.7 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 dan como resultado una correcta construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar esos pasos, y cuánto esfuerzo, tiempo y recursos se van a requerir. Cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de

prueba, la ejecución de las pruebas, la agrupación y evaluación de los datos resultantes. Una estrategia de prueba del software debe ser suficientemente flexible para promover la creatividad y la adaptabilidad necesaria para adecuar la prueba a todos los grandes sistemas basados en software. (Pressman, 2002)

1.3.8 Niveles de prueba

Las pruebas se aplican con objetivos específicos, en diferentes escenarios o niveles de trabajo. Teniendo en cuenta esto, las pruebas se agrupan por niveles de acuerdo a las diferentes etapas del proceso de desarrollo. A continuación se describen los niveles de prueba:

- **Prueba de Unidad:** Se centra en el proceso de verificación en la menor unidad del diseño del software: el componente o módulo. (Pressman, 2002)

De modo que una prueba de unidad o una prueba unitaria, es una forma de probar el correcto funcionamiento de un módulo de código. Esta sirve para asegurar que cada uno de ellos funcione correctamente por separado.

- **Prueba del sistema:** Las pruebas de sistema buscan discrepancias entre el programa y sus objetivos o requerimientos, enfocándose en los errores hechos durante la transición del proceso al diseñar la especificación funcional. Esto hace a las pruebas de sistema un proceso vital, ya que en términos del producto, número de errores hechos, y severidad de esos errores, es un paso en el ciclo de desarrollo generalmente propenso a la mayoría de los errores. Las pruebas de sistema tienen como objetivo ejercitar profundamente el sistema comprobando la integración de la información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica. (I. Jacobson, 2000)

En este nivel son aplicados de acuerdo a una especificación tipos de pruebas como:

- *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.

- *Prueba de seguridad:* Consisten en verificar los mecanismos de control de acceso al sistema para evitar alteraciones indebidas en los datos.

- *Prueba de resistencia:* Consisten en comprobar el funcionamiento del sistema en el umbral límite de los recursos, sometiéndole a cargas masivas. El objetivo es establecer los puntos extremos en los cuales el sistema empieza a operar por debajo de los requisitos establecidos.

- *Prueba de rendimiento:* Consisten en determinar que los tiempos de respuesta están dentro de los intervalos establecidos en las especificaciones del sistema.

- *Prueba de funcionalidad:* Pruebas fijando su atención en la validación de las funciones, métodos, servicios y casos de uso.

- *Prueba de Usabilidad:* Prueba enfocada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente de documentación de usuarios y materiales de entrenamiento.

- *Pruebas de soportabilidad:* prueba enfocada a asegurar la instalación y funcionamiento en diferentes configuraciones de hardware y software.

- *Prueba de stress:* se proponen encontrar errores debidos a recursos bajos o completitud de recursos. El objetivo de esta prueba es investigar el comportamiento del sistema bajo condiciones que sobrecargan sus recursos. Se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

- **Prueba de validación:** Son las pruebas realizadas sobre un software completamente integrado para evaluar el cumplimiento con los requisitos especificados. (I. Jacobson, 2000)

- **Prueba de integración:** El objetivo de las pruebas de integración es verificar el correcto ensamblaje entre los distintos componentes una vez que han sido probados unitariamente con el fin de comprobar que interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales especificados en las verificaciones correspondientes. (I. Jacobson, 2000)

Los tipos fundamentales de integración son los siguientes:

- *Integración incremental:* se combina el siguiente componente que se debe probar con el conjunto de componentes que ya están probados y se va incrementando progresivamente el número de componentes a probar.

- *Integración no incremental:* se prueba cada componente por separado y posteriormente se integran todos de una vez realizando las pruebas pertinentes.

➢ **Prueba de aceptación:** El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. (I. Jacobson, 2000)

La validación del sistema se consigue mediante la realización de pruebas de caja negra que demuestran la conformidad con los requisitos y que se recogen en el plan de prueba, el cual define las verificaciones a realizar y los casos de prueba asociados.

➢ **Prueba del desarrollador:** Prueba diseñada e implementada por el equipo de desarrollo. Estas pruebas pueden realizarse cruzando los programadores, analistas, u otro rol, de forma tal que no solo sea la persona la que revise su propio trabajo, pues generalmente una persona ajena es la que más errores puede detectar. Tradicionalmente estas pruebas han sido consideradas solo para la prueba de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración. Se recomienda que estas pruebas cubran más que las pruebas de unidad. (I. Jacobson, 2000)

1.3.9 Métodos de Prueba

Existen métodos de prueba independientemente de la técnica que se utilice o el nivel en que se enmarquen estas técnicas, estos métodos proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas, agrupándose en:

- Método de Caja Blanca o Estructural: Se basa en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.

- Método de Caja Negra o Funcional: Se realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.

1.3.9.1 Método de prueba de caja blanca

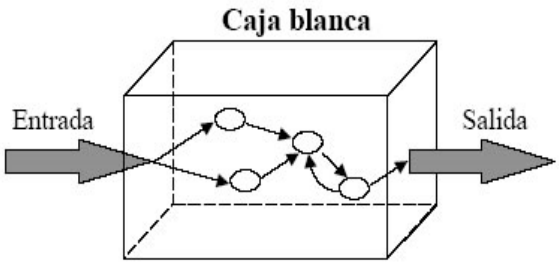

Ilustración 3. Enfoque de diseño de pruebas de caja blanca. Fuente: (Mario G. Piattini, 2004)

De acuerdo con Pressman, 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. (Pressman, 2002)

Requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código.

Mediante los métodos de prueba de la caja blanca, el ingeniero de software puede obtener casos de prueba que garanticen que:

1) Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
2) Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas.
3) Ejecuten todos los bucles en sus límites y con sus límites operacionales.
4) Se ejerciten las estructuras internas de datos para asegurar su validez. (Pressman, 2002)

La prueba de la caja blanca, a primera vista, podría parecer impracticable puesto que no es posible aplicarla exhaustivamente para grandes sistemas, sin embargo, no se debe desechar; ya que se puede

elegir y ejercitar una serie de caminos lógicos importantes, que invoquen además las estructuras de datos más importantes para comprobar su validez.

Técnicas de prueba de caja blanca

A continuación se describen varias técnicas de caja blanca mediante las cuales se realizan los casos de prueba:

- **Prueba del camino básico**

Es una técnica que 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. Aspectos claves de este tema: (Pressman, 2002)

- *Grafo de flujo o grafo del programa:* Representa el flujo de control lógico de un programa y se utiliza para trazar más fácilmente los caminos de éste. (Cada nodo representa una o más sentencias procedimentales y cada arista representa el flujo de control).

Para construir el grafo se debe tener en cuenta la notación para las instrucciones.

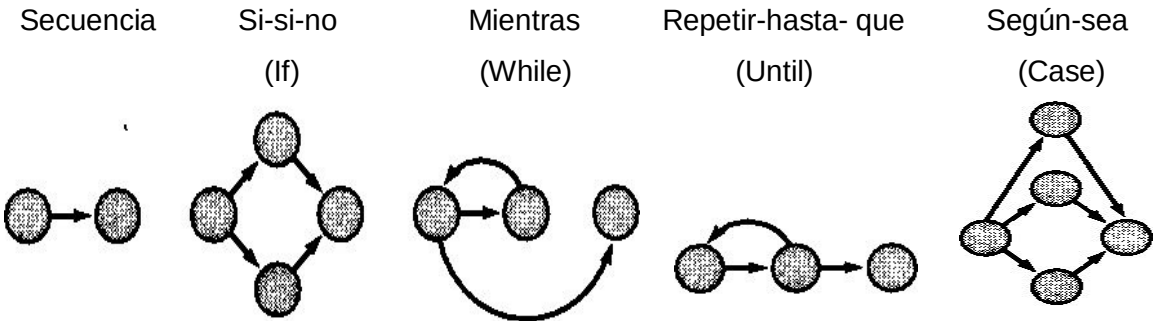

Ilustración 4. Notación de grafos de flujo para las instrucciones: Secuenciales, If, While

- *Complejidad ciclomática: Es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto de las pruebas, el cálculo de la complejidad ciclomática representa el número de caminos independientes del conjunto básico de un*

programa. Esta medida ofrece al probador de software un límite superior para el número de pruebas que debe realizar para garantizar que se ejecutan por lo menos una vez cada sentencia.

Se calcula:

1ra vía V (G)= Número de Regiones
2da vía V (G)= Número de Aristas – Número de Nodos + 2
3ra vía V (G)= Número de Nodos Predicados +1

- *Camino independiente:* Cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición.

➢ **Prueba de la estructura de control**

Dentro de este tipo de prueba se contempla el método del camino básico mencionado anteriormente pero además existen otras pruebas asociadas que permiten ampliar la cobertura de la prueba y mejorar su calidad. Estas son: (Pressman, 2002)

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

- *Condición simple: es una variable lógica o una expresión relacional (E1 < operador - relacional > E2).*

- *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.

➢ **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. (Pressman, 2002)

➢ **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). (Pressman, 2002)

1.3.9.2 Método de prueba de caja negra

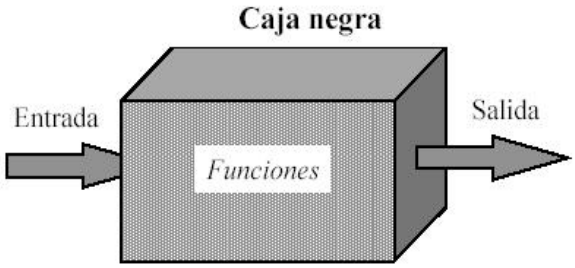

Ilustración 5. Enfoque de diseño de pruebas de caja negra. Fuente: (Mario G. Piattini, 2004)

Conociendo la función específica para la que se diseñó el producto, se puede llevar a cabo pruebas que demuestren que cada una de las funciones es totalmente operativa y a la vez buscar errores en cada función, este tipo de prueba se le llama prueba de caja negra.

Para Pressman las pruebas de caja negra se centran en los requisitos funcionales del software, es decir, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Se trata de un enfoque que intenta descubrir diferentes tipos de errores que no se encuentran con los métodos de caja blanca. (Pressman, 2002)

La prueba de caja negra para este autor, intenta encontrar errores de las siguientes categorías:

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

Desde un punto de vista parecido la prueba funcional o de caja negra se centra en el estudio de la especificación del software, del análisis de las funciones que debe realizar, de las entradas y de las salidas. (Mario G. Piattini, 2004)

Para Glenford el objetivo de las pruebas de caja negra es estar totalmente despreocupado del comportamiento interno y de la estructura del programa. En lugar de esto, concentrarse en encontrar las circunstancias en las cuales el programa no se comporte según su especificación. En este acercamiento, los datos de prueba se derivan solamente de la especificación. (Myers, 2004)

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

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

Técnicas de prueba de caja negra

A continuación se describen varias técnicas de prueba de caja negra mediante las cuales se realizan los casos de prueba:

- **Prueba basada en grafos**

En esta técnica se deben 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. En este método:

1. Se crea un grafo de objetos importantes y sus relaciones.
2. Se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores. (Pressman, 2002)

- **Partición equivalente**

Pressman presenta la partición equivalente como un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. 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. (Pressman, 2002)

El objetivo de la técnica 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. (Patton, 2005)

En el diseño de casos de prueba para la 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. (Kit, 1994)
Es de notar que dos tipos de clases de equivalencia están identificadas: las clases de equivalencia válidas representan entradas válidas 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). (Kit, 1994)

Se define los casos de prueba. El segundo paso es el uso de las clases de equivalencia para identificar los casos de prueba, proceso al cual se le 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; 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. (Kit, 1994)

- **Análisis de valores límite**

Los errores tienden a darse más en los límites del campo de entrada que en el centro, por ello, se ha desarrollado el análisis de valores límite (AVL) como técnica de prueba, el que lleva a una elección de casos de prueba que ejerciten los valores límite.

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. (Pressman, 2002)

Condiciones sublímite. Las condiciones limite 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 sublímite 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 debe incluir los valores en la partición inválida justo «debajo de» y «encima de» esos caracteres de la tabla ASCII, [, " y {. (Patton, 2005)

- **Prueba de la tabla ortogonal**

La prueba de la tabla ortogonal puede aplicarse a problemas en que el dominio de entrada es relativamente pequeño pero demasiado grande para posibilitar pruebas exhaustivas. La técnica de prueba de la tabla ortogonal es particularmente útil al encontrar errores asociados con fallos localizados, -una categoría de error asociada con defectos de la lógica dentro de un componente software-. (Myers, 2004)

- **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.

La idea básica es enumerar una lista de errores posibles o de situaciones propensas a error y después escribir los casos de prueba basados en la lista. Por ejemplo, la presencia del valor 0 en la entrada de un programa es una situación con tendencia a error. Por lo tanto, puede ser que se escriban los casos de prueba para los cuales los valores particulares de la entrada tienen valor 0 y para qué valores particulares de la salida se colocan de manera forzada a 0. (Kit, 1994)

1.3.10 Soluciones existentes

Durante el estudio de la investigación se verificó que no existen soluciones de un proceso de pruebas realizadas al Módulo Búsqueda Referativa. Con el diseño y aplicación de este proceso se obtendrá en su primera versión los defectos encontrados en dicho módulo, estos serán documentados y eliminados, comprobando que el módulo realice los requerimientos especificados por el cliente correctamente.

1.4 Conclusiones

El análisis realizado en este capítulo ha mostrado de manera clara, la importancia de la calidad del software, a lo que contribuyó el abarcamiento de los temas tratados, como lo relacionado al proceso de pruebas en un software, de los niveles en que las pruebas se dividen, así como de los métodos, los tipos y las técnicas que se emplean para realizarlas. Además, de la metodología y artefactos que rigen un adecuado desarrollo de software.

Capítulo 2 : Diseño y aplicación de las pruebas

2.1 Introducción

El proceso de pruebas de software debe formar parte de un proceso definido, documentado y medido para poder ser gestionado. En el presente capítulo se elaboran diferentes artefactos de prueba propuestos por el Proceso Unificado de Rational (RUP), necesarios para dirigir y organizar todas las actividades involucradas en este flujo de trabajo.

2.2 Características a probar

Las características a evaluar en el Módulo Búsqueda Referativa con la realización de las pruebas son:

- ✓ Sí se cuenta con la documentación detallada que corresponda con el módulo.
- ✓ Sí se puede usar correctamente y con facilidad.
- ✓ Sí la aplicación cumple con las funcionalidades requeridas desde el comienzo.
- ✓ Sí es comprensible para el usuario.

2.3 Plan de Prueba

Dicho plan está diseñado para asegurar que se satisfacen todos los requisitos funcionales especificados por el usuario teniendo en cuenta también los requisitos no funcionales relacionados con el rendimiento, seguridad de acceso al sistema, a los datos y procesos, así como a los distintos recursos del sistema.
Se incluyen las especificaciones de Software y Hardware, teniendo en cuenta las herramientas y los dispositivos que se necesitan para el desarrollo del sistema.

2.3.1 Especificaciones de Software y Hardware

Especificaciones de Software y Hardware

La aplicación debe de ser multiplataforma ya que debe ser compatible con los Sistemas Operativos: Windows 2000 NT, Windows XP y Linux, Ubuntu 7.10. A continuación se muestran algunos de los requerimientos de software como de hardware de dicha aplicación.

Requerimientos de Hardware:

Las computadoras que utilizarán el software a desarrollar deberán tener 64 MB de Memoria tipo RAM como mínimo.

Requerimientos de Software:

Las computadoras que utilizarán el software deben tener instalado:

- Windows 2000 NT, Windows XP Profesional ó GNU/Linux en cualquier distribución.
- Navegador Web compatible con IE 4.0 o superior, Mozilla, Opera.

El nodo (PC) que alojará la aplicación y la base de datos deberá tener instalado un servidor Apache con versión 5 de PHP, framework Symfony y PostgreSQL versión 8.3.

2.3.2 Requisitos Funcionales y Casos de Uso

Es objetivo primordial verificar en el proceso de pruebas, como parte del plan de prueba, que los requerimientos funcionales a desarrollar sean alcanzados, pues dichos requerimientos serán la base de los casos de uso del sistema y constituirán las principales funcionalidades con las que debe contar el sistema. Los requerimientos funcionales del Módulo Búsqueda Referativa se detallan a continuación:

Requerimientos funcionales que se deben probar

RF 1: Gestionar Documento
RF 2: Gestionar Préstamo
RF 3: Mostrar Reporte
RF 4: Buscar Documentos por categorías
RF 5: Generar documentos
RF 6: Búsqueda avanzada

CU 1: Caso de Uso: Gestionar Documento

Contiene una descripción de las actividades a realizar para Gestionar Documento en el sistema. El caso de uso se inicia cuando el administrador accede a la opción Documentos en el menú Gestión, para realizar las acciones de adicionar, modificar, eliminar, ver detalles y guardar en PDF un documento. El caso de uso termina cuando el sistema realiza la operación elegida, y el administrador selecciona otra opción o sale de la aplicación.

CU 2: Caso de Uso: Gestionar Préstamo

Contiene una descripción de las actividades a realizar para Gestionar Préstamo en el sistema. El caso de uso se inicia cuando el administrador accede a la opción Préstamo en el menú Gestión, con el propósito de realizar las acciones de insertar, modificar, eliminar, ver detalles o devolver un préstamo de un documento. Una vez hecha cualquiera de estas operaciones el sistema se actualiza y finaliza el caso de uso.

CU 3: Caso de Uso: Mostrar Reporte

Contiene una descripción de las actividades a realizar para Mostrar Reporte al sistema. El caso de uso se inicia cuando el administrador accede en el menú “Reportes” a cualquiera de las siguientes opciones: Entidades con deudas, Estadísticas del Archivo y Préstamos Pendientes; con el propósito de ver los detalles de los documentos obtenidos en los resultados de la búsqueda que realiza el sistema en cada una de estas acciones. El CU finaliza cuando el administrador realiza dicha operación y accede a otra opción o sale de la aplicación.

CU 4: Buscar Documentos por categorías

Contiene una descripción de las actividades a realizar para Buscar Documentos por categorías en el sistema. El CU inicia cuando el usuario desea realizar la búsqueda de un documento por cualquiera de las siguientes categorías: texto completo, autores, regiones, fecha de entrada y materias primas. Una vez mostrados los resultados de la búsqueda seleccionada el CU termina.

CU 5: Caso de Uso: Generar documentos

Contiene una descripción de las actividades a realizar para Generar documentos en el sistema. El CU inicia cuando cualquiera de los actores selecciona la opción de generar documentos en el formato que desee (pdf, word, excel). El sistema genera el documento con los datos seleccionados y finaliza el CU.

CU 6: Caso de Uso: Búsqueda avanzada

Contiene una descripción de las actividades a realizar para Búsqueda avanzada en el sistema. El CU inicia cuando el usuario desea consultar documentos dados uno o varios criterios de búsqueda: generales, autores, fecha, materia prima, provincias, yacimientos, grado de estudio. Una vez seleccionados los criterios y especificados los datos en cada uno de ellos; el sistema muestra el resultado, dando al usuario la posibilidad de realizar las acciones de ver detalles o generar documentos y finaliza el CU.

2.3.3 Casos de Prueba

El objetivo principal de aplicar pruebas al Módulo Búsqueda Referativa es ejecutarlo teniendo en cuenta un conjunto de casos de prueba, con la primordial intención de detectar fallas o errores que puedan existir y corregirlos, de esta manera, darle más consistencia y una mayor confiabilidad.
Para la elaboración de los casos de prueba y su posterior aplicación hay que tener claro que, un buen caso de prueba es aquel que tiene altísima probabilidad de mostrar un error que hasta entonces no se había descubierto y que por tanto los desarrolladores puedan corregirlo.

Los casos de prueba deben verificar:

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

Para la realización de las pruebas se diseñaron 6 casos de prueba, un caso de prueba por cada caso de uso.
A continuación se muestra el nombre de los casos de prueba:
CP 1: Gestionar Documento.
CP 2: Gestionar Préstamo.
CP 3: Mostrar Reporte.
CP 4: Documentos por categorías.
CP 5: Generar documentos.
CP 6: Búsqueda avanzada

2.4 Configuración del entorno de prueba

La configuración del entorno donde se vayan a ejecutar las diferentes pruebas que se realizan a un software es un aspecto muy importante dentro del proceso de pruebas, pues si no se analizan bien los recursos de software y hardware que necesita el producto que se está construyendo, a la hora de probar dicho producto se prescindirá de los elementos necesarios para la ejecución de un proceso de pruebas exitoso. Por ello se tuvo en cuenta a la hora de llevar a cabo todo el proceso de pruebas al Módulo

Búsqueda Referativa, algunos requerimientos de hardware y de software que hicieron posible un mejor desarrollo de las pruebas.

Algunos de los requerimientos que se consideraron necesarios para las pruebas son los referenciados a continuación:

Requerimientos de Software:

- Windows 2000 NT, Windows XP y Linux, Ubuntu 7.x.
- Sistema de Gestión de base de datos PostgreSQL versión 8.3.
- Servidor Web: Apache.
- Microsoft Office 2003 ó superior.

Requerimientos de Hardware:

- PC con Microprocesador Pentium III a 2.41 GHz como mínimo.
- Con 80 GB de Disco Duro como mínimo.
- 248 MB de Memoria RAM como mínimo.

2.5 Ejecución de las Pruebas

Para la realización de las pruebas se planificó el tiempo en que se realizarían, los recursos necesarios, artefactos y entregables, entre otros aspectos de vital importancia.

2.5.1 Pruebas a la documentación

La prueba de documentación se puede enfocar en dos fases. La primera fase, la revisión e inspección, examina el documento para comprobar la claridad editorial. La segunda fase, la prueba en vivo, utiliza la documentación junto al uso del programa real". (Pressman, 2002)

Dentro de las pruebas realizadas al Módulo Búsqueda Referativa tuvo lugar la prueba de documentación, esta se realizó por fases, tal y como describió Pressman: primero se revisó la documentación desde el punto de vista de redacción, ortografía y concordancia, entre otros aspectos medidos; y posteriormente se revisó la relación de ésta con la aplicación, verificando que todo lo plasmado en ella estuviera presente en la aplicación, y viceversa, que todas las funcionalidades del sistema estuvieran descritas en la

documentación. A continuación se muestra el nombre del documento implicado en la prueba, junto a una breve descripción del mismo:

- **Especificación de Requisitos**

Este es uno de los principales documentos, por contener las funcionalidades que debe realizar el sistema, o sea, los requerimientos acordados con el cliente, tanto funcionales, como no funcionales.

2.5.2 Pruebas de Seguridad

Cualquier sistema basado en computadora que maneje información sensible o lleve a cabo acciones que puedan perjudicar (o beneficiar) impropiamente a las personas es un posible objetivo para entradas impropias o ilegales al sistema. Este acceso al sistema incluye un amplio rango de actividades: piratas informáticos que intentan entrar en los sistemas por deporte, empleados disgustados que intentan penetrar por venganza e individuos deshonestos que intentan penetrar para obtener ganancias personales ilícitas. (Pressman, 2002)

La prueba de seguridad intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán, de hecho, de accesos impropios. Para citar a Beizer: ¨*Por supuesto, la seguridad del sistema debe ser probada en su invulnerabilidad frente a un ataque frontal, pero también debe probarse en su invulnerabilidad a ataques por los flancos o por la retaguardia*¨. (Beizer, 1984)

Estas pruebas necesitan tiempo y herramientas para su correcta ejecución. Para las pruebas de seguridad realizadas al Módulo Búsqueda Referativa no se utilizaron herramientas, se realizaron manualmente, en dos fases y a nivel de usuario, la primera para probar la seguridad que brinda el sistema a la hora de autenticarse, donde se verificó que solos los actores con acceso al sistema están habilitados para accederla; y la segunda para probar la seguridad que brinda el sistema a sus funcionalidades, donde se verificó que un actor solamente tuviera acceso a las funciones y datos que su usuario tiene permitido.

El módulo cuenta con dos tipos de usuarios, mostrados a continuación:

- **Usuario o invitado:** solamente realiza las funciones básicas: buscar documentos.
- **Administrador:** accede a todas las funcionalidades del módulo.

2.5.3 Pruebas de Estructura

Con el objetivo de verificar que todos los enlaces (detenidos o activos) están conectados correctamente. Se realiza en aplicaciones habilitadas para web y garantiza que se muestre el contenido adecuado según el enlace. Se hace una comprobación de que no hay enlaces rotos. Estos pueden estar rotos por varios motivos: el desplazamiento, la eliminación o el cambio de nombre de los archivos, cambio de contenido o de destino. Los enlaces también pueden estar rotos debido a un uso incorrecto de la sintaxis: los dos puntos, las barras inclinadas o las letras que falten.

2.5.4 Pruebas Funcionales

Las pruebas funcionales se realizaron para verificar que el sistema implementa adecuadamente cada uno de los requisitos acordados para el software y que funcionan correctamente. Haciendo uso de los casos de prueba diseñados, donde se recogen los escenarios correspondientes a los casos de uso del sistema y las clases de datos utilizadas para probar estos escenarios, se probó cada funcionalidad comparando los resultados obtenidos con los resultados esperados, estos últimos también contenidos en los casos de prueba. De esta manera se verificó que el sistema respondía a las necesidades del cliente de la misma manera que se solicitó. Se revisó también que las interfaces de la aplicación utilizarán correctamente el idioma definido, que mantuvieran concordancia entre sus textos y buena ortografía, que los mensajes mostrados fueran claros y concisos. Otro aspecto que se tuvo en cuenta fue la estética y visualización de la interfaz.

2.5.5 Diseño de casos de prueba 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.

2.5.5.1 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 fructífera la ejecución de las pruebas. Esta tabla contiene los campos:

- Nombre de la sección: Se especifica el nombre de la sección [SC 1: Nombre de la sección].
- Escenarios de la sección: Se especifican los escenarios de cada sección [EC 1.1: Nombre del Escenario].
- Descripción de la funcionalidad: Se describe brevemente la funcionalidad del escenario.

A continuación se muestra el caso de prueba de caja negra para el caso de uso Gestionar Documento.

Nombre del CU: Gestionar Documento.

Descripción General: El caso de uso se inicia cuando el administrador accede a la opción Documentos en el menú Gestión, para realizar las acciones de adicionar, modificar, eliminar, ver detalles y guardar en PDF un documento. El caso de uso termina cuando el sistema realiza la operación elegida, y el administrador selecciona otra opción o sale de la aplicación.

Condiciones de Ejecución: El usuario debe estar autenticado y poseer los permisos de administración correspondientes al módulo.

Secciones a probar en el Caso de Uso:

- Adicionar un documento.
- Modificar documento.
- Eliminar Documento.
- Detalles de un documento.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC1: Se muestran los documentos	EC 1.1: Se muestra solamente el listado de documentos.	Se debe de mostrar el listado de algunos documentos existentes en el sistema con la opción de buscar si no se ha iniciado sesión como administrador.
	EC 1.2: Se muestran los documentos.	Se ha verificado la sección del administrador como válida. Se mostrarán algunos de los documentos existentes.

SC 2: Adicionar un documento.	EC 2.1: El usuario selecciona *Adicionar*.	El sistema muestra los formularios necesarios para adicionar un documento.
	EC 2.2: Se adiciona el documento.	El sistema muestra un mensaje informando que *"Se ha insertado un documento correctamente".* El documento es adicionado, y mostrado en el listado de documentos.
	EC 2.3: Datos incorrectos	El sistema marca en rojo los campos incorrectos y muestra un mensaje de error informando que *"Existen datos incorrectos en el formulario"*.
	EC 2.4 El usuario no adiciona el documento.	El sistema cancela todo el proceso de adición del documento y redirecciona hacia la página principal de Gestionar Documento.
	EC 2.5: Actualiza la lista de los documento.	La lista de los documentos existentes en el sistema se actualiza.
SC 3: Modificar documento	EC 3.1 El administrador selecciona modificar.	El sistema muestra los formularios del documento seleccionado con los campos que puede modificar.
	EC 3.2 Se modifica el documento.	El documento es modificado y se muestra en el listado de documentos.
	EC 3.3 El administrador no modifica el documento.	El sistema cancela todo el proceso de modificación del documento y redirecciona hacia la página principal de Gestionar Documento.

	EC 3.4 Datos incorrectos.	El sistema marca en rojo los campos incorrectos y muestra un mensaje de error informando que “Existen datos incorrectos en el formulario”.
	EC 3.5: Actualiza la lista de los documento.	La lista de los documentos existentes en el sistema se actualiza.
SC 4: Eliminar Documento.	EC 4.1: El administrador selecciona eliminar.	El administrador selecciona el documento que desea eliminar, elimina el documento y el sistema actualiza el listado.
	EC 4.2: No es confirmada la eliminación del documento.	El administrador no confirma la eliminación del documento seleccionado.
SC 5: Detalles del documento.	EC 5.1: Ver detalles del documento.	El sistema muestra una ventana emergente con a los datos específicos que no se muestran inicialmente sobre los documentos mostrados.

Tabla 1. Diseño del CP del CU: Gestionar Documento

2.5.6 Diseño de casos de prueba de Caja Blanca

2.5.6.1 Utilizando la Técnica Camino Básico

En el diseño de los casos de prueba de caja blanca para los CU del Módulo Búsqueda Referativa, se siguió la técnica de diseño del camino básico, esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.

Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.

4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

A continuación se muestra el caso de prueba de caja blanca para el caso de uso Gestionar Documentos.

Caso de Prueba Gestionar Documento (Adicionar Documento)

Código

```
public function executeInsertar ()
 {
1     if ($this->getRequest () ->getMethod () == sfRequest::POST
       && $this->getRequestParameter ('btnCancelar') == null)
    {
2     $tituloFicha = $this->getRequestParameter ('txtTituloFicha');
2     $registrador = $this->getUser () ->getAttribute ('usuario');
2     $oficina = $this->getRequestParameter ('cbxOficina');
2     $idioma = $this->getRequestParameter ('cbxIdioma');
2     $fechaRegistro = $this->getRequestParameter ('txtFechaRegistro');
2     $annoElaboracion = $this->getRequestParameter ('cbxAnnoElaboracion');
2     $urldoc = $this->getRequestParameter ('txtUrlDoc');
2     $resumenDocumento = $this->getRequestParameter ('txtResumen');
2     $noInventario= $this->getRequestParameter ('txtNoInventario');
2     $pasivo = $this->getRequestParameter ('chxPasivo');
2     $bd = $this->getRequestParameter ('chxBd');
2     $prestado = $this->getRequestParameter ('chxPrestado');
2     $clasificado = $this->getRequestParameter ('chxClasificado');
2     $condiciones = $this->getRequestParameter ('chxCondTecEco');
2     $observDocumento = $this->getRequestParameter ('txtObservaciones');
2     $provincias = $this->getRequestParameter ('cbxProvinciaDoc');
2     $ejemplar = $this->getRequestParameter ('txtEjemplar');
2    $tomo = $this->getRequestParameter ('txtTomo');
2     $cantPaginas = $this->getRequestParameter ('txtCtdPag');
2     $dibujos = $this->getRequestParameter ('txtDibujos');
```

```
2     $anexos = $this->getRequestParameter ('txtAnexos');
2     $fotos = $this->getRequestParameter ('txtFotos');
2    $observLibro = $this->getRequestParameter ('txtObservacionesLibro');
2     $certificado = $this->getRequestParameter ('chxCertificado');
2     $fechaCertificado = $this->getRequestParameter ('txtFechaCertificado');
2     $actaONRM = $this->getRequestParameter ('chxActaONRM');
2     $fechaActaONRM = $this->getRequestParameter ('txtFechaActaONRM');
2     $áreas= json_decode ($this->getRequestParameter ('hdAreas'));
2     $soporteDigital = $this->getRequestParameter ('chxSoporteDigital');
2     $soportesDigitalesDoc = $this->getRequestParameter ('txtSoporteDigitalDoc');
2     $descripcionSoporte = $this->getRequestParameter ('txtDescripcionSoporte');
2     $ubicacionSoporteDigital= $this->getRequestParameter ('txtUbicacionSoporteDigital');
2     $mPrimasDoc = $this->getRequestParameter ('cbxMPrimaDoc');
2     $showMPrimas= $this->getRequestParameter ('chxShowMatPrimas');
2     $autores = $this->getRequestParameter ('cbxAutoresDoc');
2     $showPlanchetas= $this->getRequestParameter ('chxShowPlanchetas');
2     $planchetas = $this->getRequestParameter ('cbxPlanchetasDoc');
2     $showEscalas= $this->getRequestParameter ("chxShowEscalas");
2     $escalas = $this->getRequestParameter ('cbxEscalasDoc');
2     $showYacimientos= $this->getRequestParameter ('chxShowYacimientos');
2     $yacimientos= json_decode ($this->getRequestParameter ('hdYacimientos'));
2     $resultados = $this->getRequestParameter ('cbxResultadosDoc');
2         $ficha = TfichasPeer::RegistrarFicha ($tituloFicha, $urldoc, $registrador, $clasificado, $pasivo,
$prestado, $resumenDocumento, $condiciones, $bd, $annoElaboracion, $fechaRegistro, $idioma,
$observDocumento, $oficina, $noInventario);
2   TyacimientobdrPeer::insertarYacimientos ($ficha, $yacimientos);
2   TrfichaProvinciaPeer::insertarProvincias ($ficha, $provincias);
2  $this->insertarLibro ($ficha, $ejemplar, $tomo,  $cantPaginas, $dibujos, $anexos, $fotos, $observLibro);
2    $this->insertarActaONRM ($ficha, $actaONRM, $fechaActaONRM);
2   $this->insertarCertificado ($ficha, $certificado, $fechaCertificado);
```

```
2 $this->insertarSoporteDigital ($ficha, $soporteDigital, $soportesDigitalesDoc, $ubicacionSoporteDigital,
$descripcionSoporte);
3     if ($showMPrimas != NULL)
4    TrfichaMateriaprimaPeer::insertarMPrimas ($ficha, $mPrimasDoc);
5    TrfichaAutoresPeer::insertarAutores ($ficha, $autores);
6    if ($showEscalas != NULL)
7        TrfichaEscalainvPeer::insertarEscalas ($ficha, $escalas);
8     TsectoresInvestigacionPeer::insertarSectoresInvestigacion ($ficha, $áreas);
9     if (count ($resultados) > 0)
10       TrfichaResultadosPeer::insertarResultados ($ficha, $resultados);
 11       $geometry= json_decode ($this->getRequestParameter ('hdGeometryObject'));
11       $proyeccion= $this->getRequestParameter ('hdProyeccion');
11        $cadena= "MULTIPOLYGON ((";
12        for ($i = 0; $i < count ($geometry); $i++)
          {
13                $cadena.="(";
14                for ($j= 0; $j < count ($geometry [$i]); $j++)
                  {
15                    $cadena.= $geometry [$i] [$j] [0] ->x." “. $geometry [$i] [$j] [0] ->y;
16                    if ($j < count ($geometry [$i]) - 1)
17                      $cadena.= ",";
                  }
18                $cadena.= ")";
19                if ($i < count ($geometry)-1)
20                  $cadena.=",";
          }
21        $cadena.="))";
22       TfichasPeer::insertarCoordenadasFicha ($ficha, $tituloFicha, $cadena, $proyeccion);
    }
23    $this->getUser () ->setFlash ('nuevo', $ficha);
```

```
24      $this->redirect ('ficha/index');
}
```

Grafo de Flujo de Datos

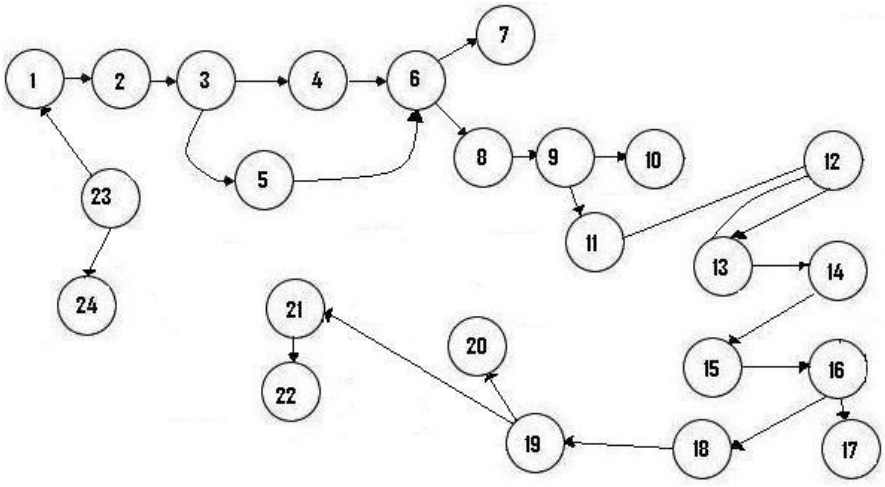

Ilustración 6. Grafo de flujo de datos para el método Gestionar Documentos (Adicionar Documento)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 26 – 24 + 2 = 4

Paso 3: Caminos básicos

CB1: 1 2 3 4 6 8 9 11 12 13 14 15 16 18 19 21 22 23 24

CB2: 1 2 3 5 6 8 9 11 12 13 14 15 16 18 19 21 22 23 24

Paso 4: Caso de prueba para el camino básico.

CB1: 1 2 3 4 6 8 9 11 12 13 14 15 16 18 19 21 22 23 24

Caso de prueba: Gestionar Documentos (Adicionar Documento).

Entrada: No. Inventario, Título, Entidad, Idioma, Fecha de Entrada, Año Elaboración, URL del documento, Resumen, Observaciones…

Resultado esperado: Se adiciona el documento en el sistema.

Resultado de la Prueba: Satisfactorio.

Caso de Prueba Gestionar Documentos (Modificar Documento)

Código

```
public function executeActualizar ()
  {
1     $idficha =$this->getRequestParameter ('hidficha');
1    $tituloFicha = $this->getRequestParameter ('txtTituloFicha');
1   $registrador =  $this->getUser () ->getAttribute ('nombre');
1  $oficina = $this->getRequestParameter ('cbxOficina');
1     $idioma = $this->getRequestParameter ('cbxIdioma');
1    $fechaRegistro = $this->getRequestParameter ('txtFechaRegistro');
1     $annoElaboracion = $this->getRequestParameter ('cbxAnnoElaboracion');
1     $urldoc = $this->getRequestParameter ('txtUrlDoc');
1     $resumenDocumento = $this->getRequestParameter ('txtResumen');
1    $NoInventario= $this->getRequestParameter ('txtNoInventario'); 1
1     $pasivo = $this->getRequestParameter ('chxPasivo');
1     $bd = $this->getRequestParameter ('chxBd');
1     $prestado = $this->getRequestParameter ('chxPrestado');
1     $clasificado = $this->getRequestParameter ('chxClasificado');
1     $condiciones = $this->getRequestParameter ('chxCondTecEco');
1     $observDocumento = $this->getRequestParameter ('txtObservaciones');
1     $provincias = $this->getRequestParameter ('cbxProvinciaDoc');
1     $ejemplar = $this->getRequestParameter ('txtEjemplar');
1     $tomo = $this->getRequestParameter ('txtTomo');
1     $cantPaginas = $this->getRequestParameter ('txtCtdPag');
1     $dibujos = $this->getRequestParameter ('txtDibujos');
1     $anexos = $this->getRequestParameter ('txtAnexos');
1     $fotos = $this->getRequestParameter ('txtFotos');
1     $observLibro = $this->getRequestParameter ('txtObservacionesLibro');
1     $certificado = $this->getRequestParameter ('chxCertificado');
```

```
1     $fechaCertificado = $this->getRequestParameter ('txtFechaCertificado');
1     $actaONRM = $this->getRequestParameter ('chxActaONRM');
1     $fechaActaONRM = $this->getRequestParameter ('txtFechaActaONRM');
1     $soporteDigital = $this->getRequestParameter ('chxSoporteDigital');
1     $soportesDigitalesDoc = $this->getRequestParameter ('cbxSoporteDigitalDoc');
1     $descripcionSoporte = $this->getRequestParameter ('txtDescripcionSoporte');
1     $ubicacionSoporteDigital= $this->getRequestParameter ('txtUbicacionSoporteDigital');
1     $matriasp= $this->getRequestParameter ('chxShowMatPrimas');
1        $mPrimasDoc = $this->getRequestParameter('cbxMPrimaDoc');
1        $autores = $this->getRequestParameter('cbxAutoresDoc');
1     $escalas = $this->getRequestParameter ('cbxEscalasDoc');
1        $escalasp= $this->getRequestParameter("chxShowEscalas");
1     $yacimientos= $this->getRequestParameter ('chxShowYacimientos');
1     $yacimientosDoc = $this->getRequestParameter ('hdYacimientos');
1     $yacimientosDoc= json_decode ($yacimientosDoc);
1     $areas= json_decode ($this->getRequestParameter ('hdAreas'));
1     $resultados = $this->getRequestParameter ('chxResultados');
1     $ficha = TfichasPeer::modificarFicha ($idficha, $tituloFicha, $urldoc, $registrador, $clasificado,
1  $pasivo, $prestado, $resumenDocumento, $condiciones, $bd, $annoElaboracion, $fechaRegistro,
1 $idioma, $observDocumento,  $oficina, $NoInventario);
1     TlibroPeer::modificarLibro ($idficha, $ejemplar, $tomo,  $cantPaginas, $dibujos, $anexos, $fotos,
1 $observLibro);
1     TrfichaProvinciaPeer::modificarProvincias ($idficha, $provincias);
1     $this->modificarActaONRM ($idficha, $actaONRM, $fechaActaONRM);
1     $this->modificarCertificado ($idficha, $certificado, $fechaCertificado);
1     $this->modificarSoporteDigital ($idficha, $soporteDigital, $soportesDigitalesDoc,
1 $ubicacionSoporteDigital, $descripcionSoporte);
1     $this->modificarMPrimas ($idficha, $matriasp, $mPrimasDoc);
1     TrfichaAutoresPeer::modificarAutores ($idficha, $autores);
1     TsectoresInvestigacionPeer::modificarAreasInvestigacion ($idficha, $áreas);
```

```
2   if ($yacimientos != null)
3     TyacimientobdrPeer::modificarYacimientosFicha ($idficha, $yacimientosDoc);
4   else
5    TyacimientobdrPeer::eliminarYacimientosFicha ($idficha);
6   if ($escalasp != NULL)
7     TrfichaEscalainvPeer::modificarEscalas ($idficha, $escalas, $escalasp);
8   else
9     TrfichaEscalainvPeer::eliminarEscalas ($idficha);
10    if (count ($resultados) > 0)
11     TrfichaResultadosPeer::modificarResultados ($idficha, $resultados);
12  else
13    TrfichaResultadosPeer::eliminarResultados ($idficha);
14      $geometry= json_decode ($this->getRequestParameter ('hdGeometryObject'));
14      $cadena= "MULTIPOLYGON (";
15      for ($i = 0; $i < count ($geometry); $i++)
        {
16              $cadena.="(";
16              $cadPolyg="(";
17              for ($j= 0; $j < count ($geometry [$i]); $j++)
                {
18                 $cadPolyg.= $geometry [$i] [$j] [0] ->x." “. $geometry [$i] [$j] [0] ->y;
19                 if ($j <  count ($geometry [$i]) - 1)
20                   $cadPolyg.= ",";
                }
21              $cadPolyg.= ")";
21              $cadPolyg.= ",". $cadPolyg;
21              $cadPolyg.= ")";
21              $cadena.= $cadPolyg;
22              if ($i < count ($geometry)-1)
23                $cadena.=",";            }
```

```
24          $cadena.=")";
24          TfichasPeer::insertarCoordenadasFicha ($ficha, $tituloFicha, $cadena);*/
24           $this->redirect ('ficha/index');
 }
```

Grafo de Flujo de datos

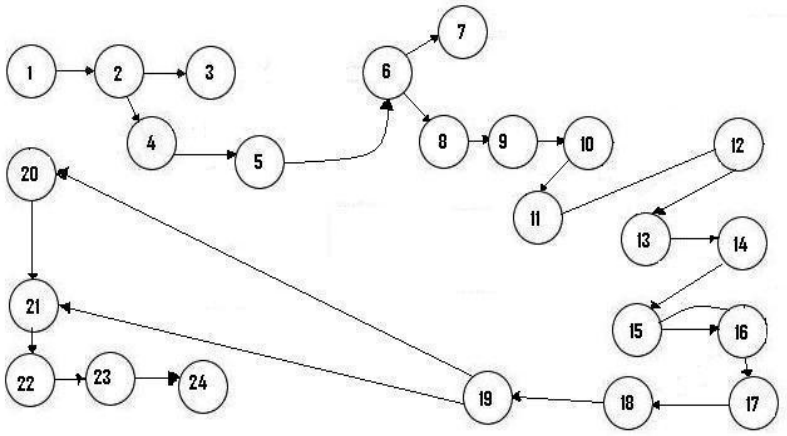

Ilustración 7. Grafo de flujo de datos para el método Gestionar Documentos (Modificar Documento)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 26 – 24 + 2 = 4

Paso 3: Caminos básicos

CB1: 1 2 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

CB2: 1 2 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 21 22 23 24

Paso 4: Caso de prueba para el camino básico.

CB2: 1 2 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 21 22 23 24

Caso de prueba: Gestionar Documentos (Modificar Documento).

Entrada: No. Inventario, Título, Entidad, Idioma, Fecha de Entrada, Año Elaboración, URL del documento, Resumen, Observaciones, Tipo de documento (pasivo, clasificado, base de datos, condiciones técnicas económicas), Tomo, Cantidad de páginas…

Resultado esperado: Se modifican los datos del documento.

Resultado de la Prueba: Satisfactorio.

Caso de Prueba Gestionar Documentos (Eliminar Documento)

Código

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

Grafo de Flujo de datos

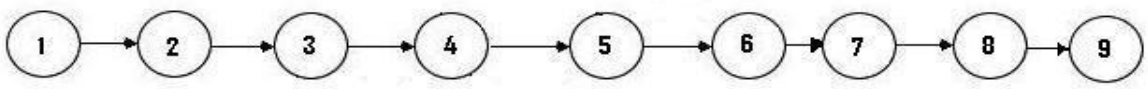

Ilustración 8. Grafo de flujo de datos para el método Gestionar Documentos (Eliminar Documento)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 11 – 9 + 2 = 4

Paso 3: Caminos básicos

CB1: 1 2 3 4 5 6 7 8 9

Paso 4: Caso de prueba para el camino básico.

CB3: 1 2 3 4 5 6 7 8 9

Caso de prueba: Gestionar Documentos (Eliminar Documento)

Entrada: No. Inventario.

Resultado esperado: Se elimina el documento.

Resultado de la Prueba: Satisfactorio.

Caso de Prueba Gestionar Documentos (Ver Detalles)

Código

```
public function executeMostrar ()
 {
   $this->id= $this->getRequestParameter ('idficha');
1        if($this->id != null)
   {
2    $this->ficha= TfichasPeer::obtenerFichaInventario ($this->id);
3    if ($this->ficha == NULL)
4      $this->redirect ('ficha/index');
4      $this->resultados= TrfichaResultadosPeer::obtenerResultados ($this->id);
4      $this->autores = TrfichaAutoresPeer::obtenerAutores ($this->id);
4      $this->yacimientos= TyacimientobdrPeer::obtenerYacimientosFicha ($this->id);
4      $this->mprimas= TrfichaMateriaprimaPeer::obtenerMateriasPrimas ($this->id);
4      $this->libro= TlibroPeer::obtenerLibro ($this->id);
4      $this->provincias= TrfichaProvinciaPeer::obtenerProvinciasFicha ($this->id);
4      $this->soportes= TrfichaSoportedPeer::obtenerSoportesDigitales ($this->id);
```

```
4       $this->certificado= TcertificadoaprobadoPeer::obtenerCertficadoInventario ($this->id);
4     $this->acta= TactaPeer::obtenerActaInventario ($this->id);
4     $this->mprimas= TrfichaMateriaprimaPeer::obtenerMateriasPrimas ($this->id);

4     $this->fareas= TsectoresInvestigacionPeer::obtenerSectoresFicha ($this->id);
4     $this->escalas= TrfichaEscalainvPeer::obtenerEscalas ($this->id);
  }
5   else
6     $this->redirect ('ficha/index');
7         }
```

Grafo de Flujo de datos

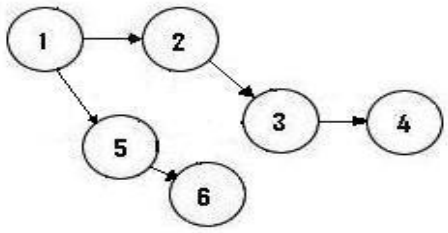

Ilustración 9. Grafo de flujo de datos para el método Gestionar Documentos (Ver Detalles)

Paso 2: Cálculo de complejidad ciclomática

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

V (G)= 8 – 7 + 2 = 3

Paso 3: Caminos básicos

CB1: 1 2 3 4 5 6 7 **CB2:** 1 2 5 6 7 **CB3:** 1 3 4 5 6 7

Paso 4: Caso de prueba para el camino básico.

CB3: 1 2 5 6 7

Caso de prueba: Gestionar Documentos (Ver Detalles)

Entrada: No. Inventario.

Resultado esperado: Se muestra el documento en una ventana emergente.

Resultado de la Prueba: Satisfactorio.

2.6 Listas de Chequeo

Se entiende por lista de chequeo (cheks-list) a un listado de preguntas, en forma de cuestionario que sirve para verificar el grado de cumplimiento de determinas reglas establecidas a priori con un fin determinado. Las preguntas, en forma de cuestionario sirven como una guía, ayuda memoria, que obliga a quien las contesta a reflexionar sobre el nivel de acatamiento de determinados requisitos (reglas). La lista de chequeo enumera una serie de ítems (muchos o pocos dependerá de la exhaustividad que se pretenda) que deberían verificarse uno a uno para asegurarnos de lograr el producto final con un nivel de calidad previamente aceptado. (Bichachi, 2002)

Para evaluar estos criterios se plantearon una serie de interrogantes que contribuyeran a la determinación de las no conformidades. Algunas de ellas se presentan a continuación:

En el documento de Especificación de requerimientos:

¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?
¿Debería especificarse algún requisito con más detalle?
¿Se han identificado los requerimientos de software y de hardware?

Usabilidad

¿Proporciona el sistema un soporte apropiado a los usuarios más novatos?
¿Se entienden la interfaz y su contenido?

Seguridad

¿Las reglas de protección de Archivos de datos, las autoridades y códigos de identificación de usuario fueron establecidas por el dueño del sistema o asignado por una autoridad más alta?
¿Toda la privacidad, la libertad de información, sensibilidad, y consideraciones de la clasificación fueron identificadas, resueltas, y establecidas?

Confiabilidad

¿Se recupera el software ante fallas?

¿La recuperación ante fallas se realiza en el tiempo requerido para la aplicación?

Portabilidad

¿Existe independencia del ambiente de software (sistema operativo, lenguaje de programación)?

2.7 Conclusiones

En este capítulo se describen las características del Módulo Búsqueda Referativa, el sistema sometido al proceso de pruebas. Para llevar a cabo el diseño y la aplicación de estas pruebas se tienen en cuenta una serie de aspectos que son abordados también en este capítulo como son:

- Elaboración del plan de prueba.
- Las pruebas se diseñan para comprobar que existen errores, no para demostrar que no existen.
- A partir de esto se determinan las características más importantes a probar.
- Las pruebas de caja negra se basan en la especificación del sistema.
- Las pruebas de código identifican casos de prueba que permiten ejecutar todos los "caminos" en un programa, asegurando que todas las sentencias se ejecutan al menos una vez.

Capítulo 3 : Resultados

3.1 Introducción

Para que un proceso de pruebas tenga éxito se requiere de un análisis final de los resultados arrojados, es decir, la evaluación del producto que se está probando de acuerdo a todos los defectos y fallos del sistema encontrados a lo largo del proceso. En el presente capítulo se presentarán algunos de los resultados obtenidos al aplicar las pruebas al módulo, se analizarán las no conformidades detectadas y la validación realizada por los expertos.

3.2 Resultados obtenidos de las pruebas aplicadas al Módulo Búsqueda Referativa

Los resultados obtenidos del proceso de prueba aplicado al Módulo Búsqueda Referativa de manera general se clasifican como satisfactorios, se encontraron algunas fallas o defectos en la aplicación del diseño de casos de prueba realizados y pruebas a la documentación de dicho módulo. A continuación se ponen ejemplos de estos defectos encontrados en la primera etapa de aplicación de las pruebas.

3.2.1 Caso de Prueba Gestionar Préstamo

SC1: Se muestran los préstamos

Escenario	Respuesta del Sistema	Resultado de la Prueb	Flujo Central
EC 1.2: Se muestran los préstamos.	El sistema muestra el listado con los préstamos existentes en el sistema.	Satisfactorio.	- Inicio - Búsqueda Referativa

Tabla 2. Resultados del CP: Gestionar Préstamo. SC 1: Se muestran los préstamos

SC 2: Adicionar un préstamo

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central

EC 2.1: El administrador selecciona la pestaña Adicionar.	El sistema muestra el formulario necesario para adicionar un préstamo.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar Préstamo
EC 2.2: Se adiciona el préstamo.	El sistema adiciona el nuevo préstamo y actualiza el listado de los préstamos y restablece los valores por defectos de los campos de la interfaz.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar Préstamo - Registrar
EC 2.3: Datos incorrectos.	El sistema da un mensaje de error y no adiciona el préstamo.	No Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar Préstamo - Registrar

EC 2.4 El usuario no adiciona el préstamo.	El sistema regresa al listado de préstamos.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar Préstamo
EC 2.5: Actualiza la lista de los préstamos.	El sistema actualiza el listado de los préstamos existentes en el sistema.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar Préstamo

Tabla 3. Resultados del CP: Gestionar Préstamo. SC 2: Adicionar préstamo

SC 3: Modificar préstamo

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 3.1 El administrador selecciona modificar.	El sistema muestra el formulario necesario para modificar un préstamo.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Modificar Préstamo

EC 3.2 Se modifica el préstamo.	El sistema modifica el préstamo y actualiza el listado de los préstamos y restablece los valores por defectos de los campos de la interfaz.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Modificar Préstamo - Modificar
EC 3.3 Datos incorrectos.	El sistema da un mensaje de error y no modifica el préstamo.	No Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Modificar Préstamo - Modificar
EC 3.4 El administrador no modifica el préstamo.	El sistema regresa al listado de préstamos.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Modificar Préstamo - Cancelar
EC 3.5: Actualiza la lista de los préstamos.	El sistema actualiza el listado de los préstamos existentes en el sistema.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos

Tabla 4. Resultados del CP: Gestionar Préstamo. SC 3: Modificar préstamo

SC 4: Eliminar préstamo

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 4.1: El administrador selecciona eliminar.	Se elimina el préstamo seleccionado y se actualiza el listado de préstamos existentes.	No Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Eliminar
EC 4.2: No es confirmada la eliminación del préstamo.	Se cierra la interfaz de confirmación de eliminación y no se elimina el préstamo seleccionado.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Eliminar - Cancelar

Tabla 5. Resultados del CP: Gestionar Préstamo. SC 4: Eliminar préstamo

SC 5: Detalles del préstamo

Escenario	Respuesta del Sistema	Resultado de la Prue	Flujo Central
EC 5.1: Ver detalles del préstamo.	El sistema muestra una ventana emergente con a los datos específicos que no se muestran inicialmente sobre los préstamos mostrados.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Detalles préstamos

Tabla 6. Resultados del CP: Gestionar Préstamo. SC 5: Detalles del préstamo

SC 6: Devolver préstamo

Escenario	Respuesta del Sistema	Resultado de la Prueb	Flujo Centra
EC 6.1: El administrador selecciona devolver documento.	Se devuelve el préstamo y el sistema actualiza el listado de préstamos existentes en el sistema.	No Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Devolver préstamo.
EC 6.2: El administrador no devuelve el documento.	El sistema regresa al listado de préstamos.	Satisfactorio.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Devolver préstamo. - Cancelar

Tabla 7. Resultados del CP: Gestionar Préstamo. SC 6: Devolver préstamo

3.3 Errores detectados en el documento Especificación de Requerimientos

Una vez registradas todas las no conformidades encontradas durante la revisión de la documentación, se tomaron en cuenta varios aspectos para la revisión del documento de Especificación de Requerimientos entre los que se encuentran:

- ✓ Ortografía.
- ✓ Formato.
- ✓ Redacción.

Como resultado de lo antes expuesto se puede afirmar que se detectaron dificultades en dicho documento, señalando que el sistema no tiene un soporte para los usuarios novatos y rectificar que ya no es polo de Geoinformática si no Centro de Geoinformática y Señales Digitales (GEySED). Hay que

destacar que todo está bien explicado y sin faltas de ortografías, en un lenguaje técnico y a la vez entendible para los usuarios de la aplicación.

A continuación se muestra una tabla con los defectos encontrados en la documentación:

	No	No conformidad	Aspecto correspondiente	Etapa	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo de Desarrollo
Documento Especificación de Requerimientos	1	El sistema no tiene un soporte para usuarios novatos.	Pruebas a la Documentación	1ra etapa	X		Se debe de poner a funcionar la ayuda del sistema para que brinde información a los usuarios novatos.	PD-Pendiente	
Documento Especificación de Requerimientos	2	Especificar que nos es Polo sino Centro de GeoInformática y Señales Digitales.	Pruebas a la Documentación	1ra etapa	X		Cambiar de Polo a Centro de GeoInformática y Señales Digitales.	PD-Pendiente	

Tabla 8. No Conformidades encontradas en la documentación.

3.4 Errores detectados en el proceso de pruebas

Se detectaron un total de 18 No Conformidades al módulo en la aplicación (Ver anexos). Además, se realizaron 2 señalamientos mediante la Lista de Chequeo al documento Especificación de Requerimientos, al finalizar el proceso de pruebas en general se detectaron un total de 20 No Conformidades. Dentro de los errores encontrados están los siguientes:

- Existen muchos enlaces que no están funcionando correctamente. Ejemplo de ellos es el caso del modificar, que casi ninguno funciona.

- Existen errores en la validación de los datos. En los campos de entrada de datos, se puede introducir prácticamente cualquier cadena de texto, no importa cuál sea, la aplicación las acepta, aun cuando se nota claramente que no pueden ser aceptadas por cuestiones lógicas.

- Validar los botones. Ejemplo, el botón cancelar, para cancelar la búsqueda de documentos.
- Modificar, que ya no es polo sino centro.
- No existe un soporte para usuarios novatos.

3.5 Evaluación del Módulo Búsqueda Referativa perteneciente al SGDG

Luego de efectuar el proceso de pruebas al Módulo Búsqueda Referativa, se considera objetivo realizar una evaluación del mismo atendiendo a los atributos de calidad siguientes:

- Eficiencia
- Seguridad
- Usabilidad
- Confiabilidad
- Portabilidad

Una síntesis de la evaluación se representa en el siguiente gráfico:

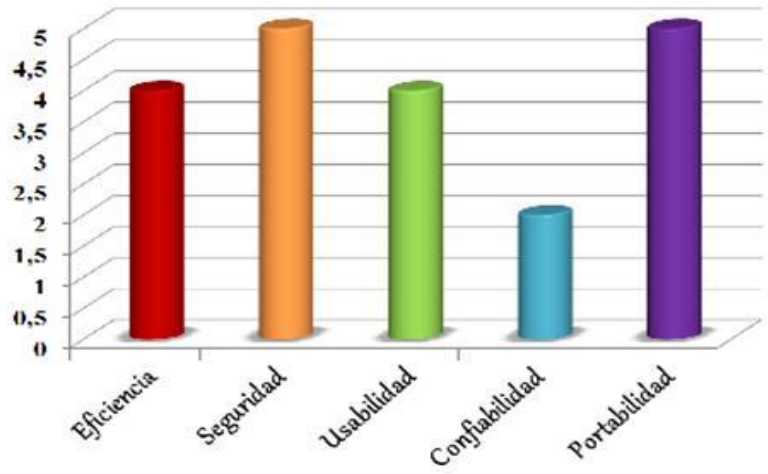

Ilustración 10. Evaluación del Módulo Búsqueda Referativa mediante atributos de calidad

La valoración se sustentó en una lista de chequeo (ver expediente de prueba). Esta lista contiene una serie de preguntas cuyas respuestas consisten en una evaluación que se le confiere a cada uno de estos

parámetros de calidad por separado. Dicha evaluación se concede en dependencia del grado de disponibilidad de las propiedades medidas, según las preguntas. Esto es, un parámetro evaluado de 0 puntos, se le atribuye a una propiedad no disponible, lo que puede significar una función o característica ausente en el sistema. Una puntuación de 2 representa una disponibilidad parcial de la propiedad, es decir, en parte son satisfactorios los resultados observados pero sigue teniendo fallos en algunos casos. En cambio una calificación de 4 describe un parámetro correctamente implementado. El 5 es para la excelencia, aquel parámetro que no tenga ningún señalamiento negativo y no presente ningún fallo.

En el caso de la Usabilidad se tuvo en cuenta un total de 16 preguntas, de las cuales 7 resultaron evaluadas de 5 puntos, 2 de 4 y 6 de 0; analizando cualitativamente estos índices se determinó que para definir una calificación final excelente (5 puntos) no se admiten aspectos evaluados con 0 ni con 2. Dicho esto, se evalúa la característica de Usabilidad del software de 4 puntos, coincidiendo este resultado con el del análisis cuantitativo en el que se promedian la suma de todos los puntos por aspectos.

En los casos particulares de Seguridad y Portabilidad, se evalúan de 5 puntos, ejerciendo el mismo análisis anterior.

Para evaluar la Eficiencia se tuvo presente el tiempo de respuesta requerido para la aplicación, siendo éste evaluado de 4 puntos.

Finalmente, se considera que el producto posee un grado de Confiabilidad óptimo, debido a que el sistema se recupera ante los fallos ocurridos, por lo que se determina una evaluación de 4 puntos para este atributo.

3.6 Validación de especialistas. Método Delphi

Entre los objetivos de la investigación, se encuentra realizar una validación externa del resultado alcanzado. Se tiene en cuenta que ningún experto conoce las identidades y respuestas de los otros expertos, para lograr así que cada uno defienda sus opiniones y en caso de ser erróneas, no habrá pérdida de su prestigio.

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 Búsqueda Referativa 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?

Mediante la aplicación de la encuesta se obtuvieron los siguientes resultados:

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

Tabla 9. Resultados de la encuesta realizada.

Como se evidencia en la tabla anterior, los expertos en sentido general consideran que la presente investigación se realizó con calidad, 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.

3.7 Conclusiones

Al finalizar este capítulo, se puede concluir que el proceso de pruebas se llevó a cabo de forma satisfactoria, pues los casos de prueba desarrollados, mostraron muchos de los errores del sistema. Estos se pueden clasificar como no catastróficos, previendo que estos puedan solucionarse de forma rápida y sin mayores problemas, pues están debidamente documentados, para una rápida comprensión por el equipo de desarrollo, y seguidamente pueda ser desplegado para pruebas de aceptación de usuarios.

Conclusiones generales

En toda la labor realizada en este trabajo de diploma al planificar y aplicar pruebas al Módulo Búsqueda Referativa se obtuvieron los siguientes resultados:

- Se obtuvo un amplio conocimiento acerca de las técnicas de pruebas existentes, la estrategia para su aplicación y los pasos a seguir.
- Se definió el plan de prueba especificando fundamentalmente el alcance, la estrategia y los recursos.
- Se diseñaron seis casos de prueba, respondiendo a la cantidad de casos de uso del sistema, donde se recogían todos los escenarios descritos en estos últimos; lo que permitió probar cada una de las funcionalidades acordadas con el cliente en forma de requerimientos.
- Se realizaron las pruebas necesarias a la documentación referente al sistema, para garantizar la entrega al cliente de una documentación libre de errores, a la altura del producto en sí.
- Se realizaron pruebas Funcionales, de Seguridad y de Estructura sobre el sistema, una vez integrados todos los elementos de éste; garantizando que el sistema tiene implementado todas las funcionalidades requeridas, que protege la información que manipula.
- Se documentaron los resultados arrojados por las distintas pruebas, lo que permitirá al equipo de desarrollo reparar las fallas encontradas, y hacer mejoras al sistema para posteriores versiones.

De forma general, el flujo de trabajo de prueba se ejecutó exitosamente partiendo de que el éxito de las pruebas se encuentra en la detección de errores. Se lograron los objetivos propuestos, que el Módulo Búsqueda Referativa a partir de las pruebas realizadas y su corrección posee una mayor calidad y un alto nivel técnico.

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:

- Que se le realicen pruebas de rendimiento, de carga y de estrés.

Referencias Bibliográficas

Ambler, Scott W. 2009. *Método de Pruebas Orientada a Objetos para el Ciclo de Vida Completo.* 2009.

Beizer, B. 1984. *Software System Testing and Quality Assurance.* S.l.: Van Nostrand Reinhold, 1984.

Beizer, Boris. 1995. *Black-Box Testing.* S.l.: Wiley, 1995.

Bichachi, Dra. Diana Susana. *El uso de las Listas de Chequeo (Chesk-List) como herramienta para controlar la calidad de la ley.* Capital Federal: Universidad del Salvador.

Blueprints, Architectural. November 1995. *The "4+1" View Model of Software Architecture.* S.l.: IEEE Software, November 1995.

Collazo, Manuel. 2003. *Técnicas de prueba del software. Estrategias de prueba del software.* S.l.: IEEE Standard Glossary of Software Engineering Terminology, 1990., 2003.

Fernández, Ing. Vladimir Martell. 2009. *Proyecto Técnico.* 2009.

I. Jacobson, G. Booch, J. Rumbaugh. 2000. *El Proceso Unificado de Desarrollo de Software.* La Habana: Addison Wesley, 2000. Vol. 1.

Kit, Edward. 1994. *"Software Testing in the real world".* S.l.: Addison-Wesley, 1994.

Kruchten, P. 2000. *The Rational Unified Process: An Introduction.* S.l.: Addison Wesley, 2000.

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

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

Myers, Glenford J. 2004. *The art of software Testing.* Second edition. S.l.: Wiley, 2004.

Patton, Ron. 2005. *Software Testing.* S.l.: Sams Publishing, 2005.

Pressman, Roger. 2004. *Ingeniería del Software. Un enfoque práctico.* Sexta edición. 2004.

Pressman, Roger S. 2002. *Ingeniería del software, un enfoque práctico.* Quinta edition. 2002.

Ronquillo, Ing. Dayris Espinosa. 2009. *Arquitectura de Software.* La Habana: s.n., 2009.

Rubio, Gabriel Buades. 2002. Calidad en Ingeniería de Software. *Ingeniería del Software III.* [En línea] 2002. http://dmi.uib.es/~bbuades/calidad/index.htm.

Software, Corporation Rational. 1998. *Rational Unified Process. Best Practices for Software Development Teams.* 1998.

Soto, Prof. Lauro. 2007. Proceso de Desarrollo Unificado. [En línea] 2007. http://www.mitecnologico.com/Main/ProcesoDeDesarrolloUnificado.

Bibliografía

Ambler, Scott W. 2009. *Método de Pruebas Orientada a Objetos para el Ciclo de Vida Completo.* 2009.

Beizer, B. 1984. *Software System Testing and Quality Assurance.* S.l.: Van Nostrand Reinhold, 1984.

Beizer, Boris. 1995. *Black-Box Testing.* S.l.: Wiley, 1995.

Bichachi, Dra. Diana Susana. *El uso de las Listas de Chequeo (Chesk-List) como herramienta para controlar la calidad de la ley.* Capital Federal: Universidad del Salvador.

Blueprints, Architectural. November 1995. *The "4+1" View Model of Software Architecture.* S.l.: IEEE Software, November 1995.

Collazo, Manuel. 2003. *Técnicas de prueba del software. Estrategias de prueba del software.* S.l.: IEEE Standard Glossary of Software Engineering Terminology, 1990., 2003.

Fernández, Ing. Vladimir Martell. 2009. *Proyecto Técnico.* 2009.

I. Jacobson, G. Booch, J. Rumbaugh. 2000. *El Proceso Unificado de Desarrollo de Software.* La Habana: Addison Wesley, 2000. Vol. 1.

Kit, Edward. 1994. *"Software Testing in the real world".* S.l.: Addison-Wesley, 1994.

Kruchten, P. 2000. *The Rational Unified Process: An Introduction.* S.l.: Addison Wesley, 2000.

Lesdey Saavedra López, Yohandris Pupo Blez. 2007. *Diseño y aplicación de pruebas al producto Registro, Cubano de Discapacitados.* Ciudad de la Habana: s.n., 2007.

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

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

Myers, Glenford J. 2004. *The art of software Testing.* Second edition. S.l.: Wiley, 2004.

Natalia Juristo, Ana M. Moreno, Sira Vegas. 17 de octubre de 2006. *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* 17 de octubre de 2006.

Patton, Ron. 2005. *Software Testing.* S.l.: Sams Publishing, 2005.

Phadke, M.S. 1997. *Planning efficient software tests.* S.l.: Crosstalk, 1997.

Pressman, Roger. 2004. *Ingeniería del Software. Un enfoque práctico.* Sexta edición. 2004.

Pressman, Roger S. 2002. *Ingeniería del software, un enfoque práctico.* Quinta edition. 2002.

Ronquillo, Ing. Dayris Espinosa. 2009. *Arquitectura de Software.* La Habana: s.n., 2009.

Rubio, Gabriel Buades. 2002. Calidad en Ingeniería de Software. *Ingeniería del Software III.* [En línea] 2002. http://dmi.uib.es/~bbuades/calidad/index.htm.

Software, Corporation Rational. 1998. *Rational Unified Process. Best Practices for Software Development Teams.* 1998.

Soto, Prof. Lauro. 2007. Proceso de Desarrollo Unificado. [En línea] 2007. http://www.mitecnologico.com/Main/ProcesoDeDesarrolloUnificado.

Glosario de Términos

UCI: Universidad de las Ciencias Informáticas.

MINBAS: Ministerio de la Industria Básica.

ONRM: Oficina Nacional de Recursos Minerales.

SGDG: Sistema de Gestión de Datos Geológicos.

CU: Caso de Uso

CP: Caso de Prueba

GEySED: Geoinformática y Señales Digitales.

Anexos

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación.	1	Cuando en el campo del título se introducen caracteres especiales, o incorrectos, no se debería de adicionar el documento. Ejemplo de caracteres en el título: €¬>☺♦♦•◘○B.	- Inicio - Búsqueda Referativa - Gestión de documentos - Adicionar	1ra etapa	X		En el campo del título del documento poner títulos que se puedan aceptar, que no se pongan caracteres que invaliden la entrada.	PD – Pendiente	
Aplicación.	2	Cuando en el campo del nombre se introducen caracteres especiales, o incorrectos, no se debería de adicionar el préstamo. Ejemplo de caracteres en el nombre: €¬>☺♦♦•◘○*B.*	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar	1ra etapa	X		En el campo del nombre del documento poner nombres que se puedan aceptar, que no se pongan caracteres que invaliden la entrada.	PD – Pendiente	

Aplicación.	3	Cuando en el campo del primer apellido se introducen caracteres especiales, o incorrectos, no se debería de adicionar el préstamo. Ejemplo de caracteres en el primer apellido: €¬>☺♦♦•◘○B.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar - Registrar	1ra etapa	X		En el campo del primer apellido del documento poner apellidos que se puedan aceptar, que no se pongan caracteres que invaliden la entrada.	PD – Pendiente	
Aplicación.	4	Cuando en el campo del segundo apellido se introducen caracteres especiales, o incorrectos, no se debería de adicionar el préstamo. Ejemplo de caracteres en el segundo apellido: €¬>☺♦♦•◘○B.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar - Registrar	1ra etapa	X		En el campo del segundo apellido del documento poner apellidos que se puedan aceptar, que no se pongan caracteres que invaliden la entrada.	PD – Pendiente	
Aplicación.	5	Cuando en el campo del nombre se introducen números no se debería de adicionar el préstamo. Ejemplo de números en el nombre: 3455. Ejemplo de caracteres en el	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar - Registrar	1ra etapa	X		En el campo del nombre del documento poner nombres que se puedan aceptar, que no se pongan caracteres que invaliden la entrada.	PD – Pendiente	

		nombre: ♦♣◘•♦							
Aplicación.	6	Cuando en el campo del primer apellido se introducen caracteres especiales, o incorrectos, no se debería de adicionar el préstamo. Ejemplo de caracteres en el primer apellido: 22.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar - Registrar	1ra etapa	X		En el campo del primer apellido del documento poner apellidos que se puedan aceptar, que no se pongan caracteres que invaliden la entrada.	PD – Pendiente	
Aplicación.	7	Cuando en el campo del segundo apellido se introducen números no se debería de adicionar el préstamo. Ejemplo de números en el segundo apellido: 66.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Adicionar - Registrar	1ra etapa	X		En el campo del segundo apellido del documento poner apellidos que se puedan aceptar, que no se pongan caracteres que invaliden la entrada.	PD – Pendiente	
Aplicación.	8	No se muestra un botón cancelar para salir de esta interfaz en caso de que no se desee buscar documentos por categorías.	- Inicio - Búsqueda Referativa - Buscar documentos	1ra etapa	X		Colocar un botón cancelar por si el usuario no realiza la búsqueda.	PD – Pendiente	

Aplicación.	9	No se muestra un botón cancelar para salir de esta interfaz en caso de que no se desee una búsqueda avanzada de documentos.	- Inicio - Búsqueda Referativa - Buscar documentos - Búsqueda Avanzada	1ra etapa	X		Colocar un botón cancelar por si el usuario no realiza la búsqueda.	PD – Pendiente	
Aplicación.	10	No se encuentra validado para generar documentos con formato xls.	- Inicio - Búsqueda Referativa	1ra etapa	X		Validar la generación de documentos para la opción con formato excel.	PD – Pendiente	
Aplicación	11	Ver detalles de documentos no realiza ninguna acción.	- Inicio - Búsqueda Referativa - Gestión de documentos	1ra etapa	X		Implementar esta acción para poder ver algunos campos con detalles del documento que no se ven en la tabla que sale al inicio.	PD – Pendiente	
Aplicación	12	Ver detalles de documentos en la búsqueda de documentos no realiza ninguna acción.	- Inicio - Búsqueda Referativa - Buscar documentos	1ra etapa	X		Implementar esta acción para poder ver algunos campos con detalles del documento que no se ven en la tabla que sale al inicio.	PD – Pendiente	
Aplicación	13	Ver detalles de documentos	- Inicio	1ra etapa	X		Implementar esta acción	PD –	

		en la búsqueda avanzada de documentos no realiza ninguna acción.	- Búsqueda Referativa - Buscar documentos - Búsqueda Avanzada				para poder ver algunos campos con detalles del documento que no se ven en la tabla que sale al inicio.	Pendiente	
Aplicación	14	Muestra el mensaje para eliminar, y cuando se confirma lo elimina pero no actualiza la tabla.	- Inicio - Búsqueda Referativa - Gestionar Préstamos - Eliminar	1ra etapa	X			PD – Pendiente	
Aplicación	15	Cuando se le da para generar en formato word, este enlace está roto o no implementado.	- Inicio - Búsqueda Referativa - Gestionar Préstamos	1ra etapa	X		Implementar esta acción para poder ver los documentos en formato word.	PD – Pendiente	
Aplicación	16	Cuando se le da para generar en formato word, este enlace está roto o no implementado.	- Inicio - Búsqueda Referativa - Opciones de Búsqueda	1ra etapa	X		Implementar esta acción para poder ver los documentos en formato word.	PD – Pendiente	
Aplicación	17	Cuando se le da para generar en formato word, este enlace está roto o no	- Inicio - Búsqueda Referativa	1ra etapa	X		Implementar esta acción para poder ver los documentos en formato	PD – Pendiente	

		implementado.	- Gestionar Documentos				word.		
Aplicación	18	Cuando se le da para generar en formato word, este enlace está roto o no implementado.	- Inicio - Búsqueda Referativa - Opciones de Búsqueda - Avanzada	1ra etapa	X		Implementar esta acción para poder ver los documentos en formato word.	PD – Pendiente	

Tabla 10. No Conformidades encontradas.