Universidad de las Ciencias Informáticas

Facultad 9

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Sistema de Gestión de Datos Geológicos. Módulo: Registro Petrolero. Rol de Diseñador de Casos de Prueba.

Autor: ***Reynier Armando Onna Lao.***

Tutor: ***Ing. Gerdys E. Jiménez Moya.***

Ciudad de La Habana, junio 2010.

"Año 52 de la Revolución".

Declaración de Autoría

Ciudad de La Habana, junio 2010
"Año 52 de la Revolución"

Yo: **Reynier Armando Onna Lao** declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del 2010.

Reynier Armando Onna Lao	Ing. Gerdys Ernesto Jiménez Moya
_____	_____
FIRMA DEL AUTOR	**FIRMA DEL TUTOR**

Datos de Contactos

Autor

Nombre: Reynier Armando.

Apellidos: Onna Lao.

Correo electrónico: raonna@estudiantes.uci.cu

Tutor

Nombre: Gerdys Ernesto

Apellidos: Jiménez Moya.

Correo electrónico: gejimenez@uci.cu

"La responsabilidad nuestra es luchar porque la calidad del producto que aquí se haga sea de las mejores y la mejor posible..."

Ernesto Ché Guevara

Opinión del Tutor

Universidad de las Ciencias Informáticas

Tesis para optar por el título de: Ingeniero en Ciencias Informáticas

Título: Sistema de Gestión de Datos Geológicos. Módulo: Registro Petrolero. Rol de Diseñador de Casos de Pruebas.

Autor: Reynier Armando Onna Lao.

Tutor: Ing. Gerdys Ernesto Jiménez Moya.

INFORME DE TUTORÍA

Dedicatoria

En especial a mis padres José A. Onna Esparraguera y Moraima I. Lao Sierra, pues este sueño es también de ellos ya que han sido los pilares en mi educación y formación.

A mi familia desde mi abuelita Georgina hasta mi sobrino Thiago, que me han dado el apoyo incondicional que he necesitado en los momentos más difíciles.

A mis hermanos por ser un ejemplo digno a seguir.

Con mucho amor y dedicación a mi flaca linda Anisley, ya que tu amor, preocupación y apoyo han sido inmensamente grandes.

A mis amistades por saber que tienen un amigo más en quien confiar, y sin ustedes yo no soy nadie.

A nuestro Comandante en Jefe Fidel Castro Ruz y nuestro pueblo heroico, que cada día lucha por un mundo mejor.

Agradecimientos

Estoy y estaré siempre agradecido de mis padres por haberme traído al mundo con mucho amor, de haberme educado y formado siendo hoy lo que soy.

A nuestra Revolución que sin ella hubiera sido más difícil mi formación como profesional. Gracias a esta revolución hermosa que cada día nosotros rejuvenecemos y fortalecemos, para que nuestros nietos tengan la posibilidad de disfrutar lo que tuve yo y mucho más.

A todos los profesores que contribuyeron a mi formación desde el círculo infantil hasta la Universidad.

A mi tribunal de tesis en especial mi tutor Gerdys E. Jiménez Moya que supo guiarme por el camino correcto, dando su apoyo ilimitado para que yo saliera adelante además de ser como mi hermano mayor.

A mis amigos y amigas que me brindan su apoyo de todo corazón en todo lo que me ha hecho falta, por haber compartido con ellos una parte de nuestras vidas que nunca se olvida.

Agradecer a una persona en especial, que ha compartido todos estos años de la Universidad, en momentos malos y buenos, gracias por tu cariño, confianza y esfuerzo, esa persona hermosa eres tú Anisley Cedeño Escalona.

En fin para todas aquellas personas que de una forma u otra me han apoyado para llegar hasta aquí y darle a esta Revolución un hombre de bien.

Resumen

El presente trabajo tiene como objetivo fundamental desarrollar las pruebas de software al módulo Registro Petrolero 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.

Se realiza un estudio de definiciones sobre los niveles, técnicas y métodos de pruebas existentes; dentro de las pruebas a realizar se encuentran: los métodos de pruebas de caja negra y caja blanca, utilizando respectivamente las técnicas de partición equivalente y camino básico. Además, se efectúan pruebas a la documentación haciendo uso de listas de chequeo, las cuales evalúan y miden la calidad del documento para hacer entrega del mismo libre de errores.

Mediante las pruebas funcionales se verificó la correcta ejecución de los requisitos definidos por el cliente a través de las funcionalidades implementadas en el sistema y se probó el nivel de seguridad y acceso de los usuarios a la información que se manipula en el módulo. Para llevar a cabo la realización de las pruebas se planificó previamente una estrategia a seguir para guiar el proceso y finalmente se realizó la evaluación de los resultados obtenidos.

PALABRAS CLAVES

Calidad, estrategia, métodos, pruebas de software

Abstract

This work has among its main objectives to develop software testing to the module Petroleum Registry which is part of Management System Geological Data (SGDG) as a production line of Geoinformatics and Digital Signal Center (GEySED) from School 9 at the University of Computing Sciences.

Is performed a study on the definitions of levels, techniques and methods of existing evidence; within the tests to be performed are: methods of black box and white box testing, using equivalent partition and basic way techniques. Also are tested to the documentation using checklists, which evaluate and measure the quality of the document to deliver it free of errors.

Through functional tests be verified the correct execution of the requirements customer-defined through the functionalities implemented in the system and tested the level of security and user access to information that is manipulated in the module. To perform the tests realization previously planned a strategy to follow to guide the process and finally the evaluation of results.

KEYWORDS

Quality, strategy, methods, software testing

Índice de Figuras

Índice de Tablas

Índice de Contenido

Introducción

El hombre mediante el trabajo ha logrado mejorar su forma de vida, ha desarrollado la economía, la cultura, el deporte, la salud y el conocimiento científico en general. Este conocimiento aplicado en la rama de la informática ha permitido solucionar problemas de la vida cotidiana satisfaciendo las necesidades de la humanidad y al desarrollo tecnológico de la sociedad; propiciando la competencia entre las industrias del software que cada vez son más fuertes y complejas en cuanto al desarrollo de productos, estos demandan una calidad adecuada en correspondencia con los requisitos del cliente, sin embargo, en ocasiones no se obtienen los resultados que realmente se esperan, convirtiéndose en un tema exigente en la sociedad a nivel mundial.

La calidad de software obtenida en el producto implica un conjunto de cualidades que caracterizan y determinan la existencia de un software: la eficiencia, flexibilidad, confiabilidad, usabilidad, seguridad e integridad. Es medible e imprescindible tener en cuenta su obtención y control durante todas las etapas del ciclo de vida de un software. Este interés crece de forma rápida y continua a lo largo de todos estos años de avances tecnológicos. Se debe a que los clientes se hacen cada vez más selectivos y comienzan a rechazar productos poco fiables o que no dan respuesta a sus necesidades. Es decir, consiste en tratar de cambiar la idea de un simple proceso sistemático a un proceso determinado por la planificación y la metodología. (Carrasco, 1995)

Por las exigencias de los clientes es necesario y de suma importancia hacer un software con calidad, que cumpla con los requisitos definidos para que tenga buena aceptación y satisfacción por parte de los usuarios del mismo.

El sector geológico no ha quedado ajeno a este tipo de exigencias en cuanto a la calidad de los productos de software que se obtienen como estrategia para la automatización de sus procesos y actividades fundamentales por lo que implica la geociencia para el desarrollo de la sociedad, así lo corroboró el Dr. Manuel Iturralde Vinent en el marco de la Primera Convención Cubana de Ciencias de la Tierra ¨… el mundo está sumergido en un proceso de cambio de concepción en cuanto al papel que desempeñan las geociencias para el desarrollo de la vida...¨(Vinent, 2005).

La Geología es la ciencia que estudia la forma interior del globo terrestre, en cuanto a su composición, estructura, evolución, recursos minerales, proporcionando grandes avances científicos en esta rama. (Morales, 2003)

Los Sistemas de Información Geográfica y de Gestión de Datos Geológicos son productos que necesita el hombre para manejar los recursos naturales por sus distintas aplicaciones en la rama de la geología.

Cuba tiene una tradición en las geociencias, que se ha fortalecido luego del triunfo de la Revolución y ha permitido que el país exhiba, entre otros, resultados científicos en la extracción y prospección petrolera. Todos estos procesos están bajo el mando del Ministerio de la Industria Básica (MINBAS).

El MINBAS es la organización estatal cubana responsable de tres importantes sectores de la economía de nuestro país, la Geología, la Energía y la Minería, así como la Química Básica. Está formada por siete grupos empresariales QUIMEFA, CUBAPETROLEO, UNION ELECTRICA, CUBANIQUEL, GEOMINSAL, CEMENTO, GEIQ que reúnen 147 empresas con 100 000 trabajadores, de estos 14 600 poseen nivel universitario y 25 000 son mujeres. Además, tienen entidades independientes adscriptas al ministerio como el Profilactorio Nacional Obrero, la Escuela Superior de la Industria Básica y Politécnico (CNCI), Tecnomática, la Empresa de Servicios (SERVIBASICA) y la Oficina Nacional de Recursos Minerales (ONRM). (MINBAS, 2007)

La ONRM es una de las entidades pertenecientes al MINBAS y está muy relacionada con el sector geológico cubano, es la autorizada para la generación, administración y uso fundamental del conocimiento geológico, por tanto, constituye la autoridad minera del país. (ONRM, 2009)

El proyecto Sistema de Gestión de Datos Geológicos (SGDG) que se lleva a cabo como parte de las actividades del Programa Nacional de Informatización del Conocimiento Geológico (PNICG), dentro de las instancias productivas del Centro Geoinformática y Señales Digitales (GEySED), surge por la necesidad de agilizar, automatizar e informatizar la ONRM, que se responsabiliza en garantizar el aprovechamiento racional de los recursos minerales del país y ejercer con eficiencia, rigor técnico y responsabilidad el control estatal sobre la geología, minería y el petróleo.

El SGDG debe permitir manipular de forma segura la información que se genera o que se transfiere en la mencionada entidad. Brindará una solución informática para cubrir las necesidades de la ONRM y está formado por los siguientes módulos: Seguridad, Búsqueda Referativa, Registro Minero, Inventario de Minerales Sólidos, Inventario de Petróleo y Gas, Inventario de Aguas Minerales, Nomencladores y Registro Petrolero.

Situación Problemática

Registro Petrolero es el módulo donde se inscriben los datos relacionados con las actividades petroleras y las personas que las llevan a cabo. En él se mantienen actualizadas las anotaciones sobre las compañías extranjeras calificadas para la realización de actividades de exploración-producción-explotación de petróleo y gas en Cuba. Además, este realiza solicitudes de Permisos de Reconocimiento y ejecuta los actos registrales establecidos, así como, notifica y certifica los derechos otorgados. (Fernández, 2009).

Se hace necesario encontrar y documentar los defectos en la calidad del módulo Registro Petrolero del SGDG, así como, validar y aprobar las asunciones realizadas en la especificación del diseño y los requisitos en una demostración concreta en dicho módulo. El objetivo del mismo es que el producto final quede con la calidad requerida y llegue a mano de los clientes con la menor cantidad de errores posibles.

Por tanto, se identifica el siguiente **problema a resolver:** ¿Cómo lograr la entrega del módulo Registro Petrolero libre de defectos?

Para darle solución se define como **objetivo general:** Elaborar la documentación técnica del proceso de pruebas correspondiente al módulo Registro Petrolero que forma parte del Sistema de Gestión de Datos Geológicos.

El problema permite establecer como **objeto de estudio:** El proceso de pruebas de sistemas de gestión de información, enmarcando en el siguiente **campo de acción:** Diseño e implementación de las pruebas del módulo Registro Petrolero.

Se plantea la siguiente **idea a defender:** El diseño y la implementación de las pruebas al módulo Registro Petrolero que forma parte del Sistema de Gestión de Datos Geológicos garantizarán que el mismo se entregue libre de defectos.

Para darle cumplimiento al objetivo 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 Registro Petrolero.
- Diseñar los Casos de Prueba.
- Realizar las pruebas de caja blanca al módulo Registro Petrolero.
- Realizar las pruebas de caja negra al módulo Registro Petrolero.
- Caracterizar el estado de la implementación una vez finalizado el proceso de pruebas.
- Validar las pruebas realizadas.

Concluida la investigación se espera obtener como **resultado**: La evaluación y validación de la implementación del módulo Registro Petrolero a partir del desarrollo del proceso de pruebas.

Durante la investigación se utilizan varios métodos científicos:

Métodos teóricos:

- Método Analítico - Sintético: Se utiliza para caracterizar las técnicas y herramientas dentro del proceso de pruebas del módulo Registros Petroleros.
- Método Histórico – lógico: Se utiliza para investigar la información que se tiene hasta el momento sobre las pruebas y resumir los aspectos fundamentales necesarios para la investigación.

Métodos empíricos:

- Método Observación: Observar cómo funciona el sistema con el diseño de pruebas para verificar el funcionamiento de este.

La investigación está estructurada en tres capítulos, los que se definen a continuación:

Capítulo 1. Fundamentación Teórica: En este capítulo se analizan los aspectos fundamentales que están relacionados con las pruebas de software y se hacen referencias a las metodologías de desarrollo y las pruebas que se realizan para validar la calidad del módulo Registro petrolero.

Capítulo 2. Diseño y aplicación de las Pruebas al Módulo Registro Petrolero: Se enmarca en los resultados que se deben obtener en el módulo en específico. En el mismo se hará referencia a la aplicación en general y se definirá la estrategia a seguir, la configuración del entorno en que serán realizadas las pruebas. Además, se hará referencia a los procedimientos de estas, que servirán de base para el diseño y la ejecución de los casos de pruebas.

Capítulo 3. Análisis de los resultados: Se hará un análisis de los resultados obtenidos a raíz de la aplicación de las pruebas al software y se hará un resumen de los principales errores que se encontraron durante las pruebas.

Capítulo 1: Fundamentación Teórica.

Introducción

En el presente capítulo se exponen conceptos relacionados con la calidad de software, se abordan los aspectos fundamentales de la ingeniería de pruebas, los niveles de pruebas, métodos, tipos o técnicas de diseños, procedimientos, herramientas de automatización y planes de pruebas. Se hará referencia a las definiciones que propone la metodología de desarrollo seleccionada para este proceso.

1.1 Conceptos asociados al dominio del problema.

Cuando se habla respecto a la calidad en diversos ámbitos: empresa privada, organismos públicos y del estado, centros educacionales entre otros. Se basa principalmente en que la calidad desempeña un papel fundamental en la competitividad de las organizaciones, así como también se considera que los consumidores de productos y servicios ya no se conforman con lo que les ofrecen, pues modifican y aumentan sus gustos y exigencias.

La calidad del software no es directamente comparable con la calidad de manufactura de productos, además identifica al menos tres problemas básicos que son propios del desarrollo de software: atender requisitos que no se establecen directamente en la especificación del sistema, existencia de ciertas características de la calidad que son complejas de especificar en forma no ambigua y la dificultad en desarrollar un sistema que satisfaga las expectativas del usuario. (Sepúlveda, 2009)

1.1.1 ¿Qué es la calidad?

La ISO - 8402 define la calidad como el conjunto de propiedades y características de un producto o servicio, que le confieren aptitud para satisfacer unas necesidades explícitas o implícitas.

Se define también como “Grado en el que un conjunto de características inherentes cumplen con los requisitos” así lo plantea la ISO 9000 - 2000.

“La calidad del software es el grado con el que un sistema, componente o proceso cumple los requisitos especificados y las necesidades o expectativas del cliente o usuario”. (IEEE, Std. 610-1990).

Según Roger Pressman es la “Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las

características implícitas que se espera de todo software desarrollado profesionalmente". (Pressman, 2005).

La calidad de software no es más que hacer cumplir los requisitos funcionales de un sistema según las necesidades o expectativas del cliente. Si no se sigue una metodología siempre habrá falta de calidad, ya que esta es una guía para indicar paso a paso las actividades a realizar y lograr el producto final deseado por el cliente.

La calidad de software es un problema actual que afecta tanto a los productores de software como a los clientes. Sucede que muchas veces los clientes reciben el software sin haberse completado la etapa de prueba correspondiente. Debido al aumento de la informatización a nivel mundial, los desarrolladores de software en ocasiones han brindado poco interés a la calidad de sus productos.

Se hace necesario evaluar la calidad de todo software antes de ser liberado a los usuarios finales, respecto a diferentes aspectos, como son: sus requisitos de funcionalidad, confiabilidad y rendimiento. La actividad esencial en este aspecto es la prueba, esta es la que permite encontrar las fallas en el software antes de que sea puesto en manos del usuario final.

Para alcanzar buena calidad en un producto de software es esencial establecer un programa de medidas a tomar con respecto a los suministradores. Es también importante utilizar los modelos y métodos apropiados para controlar el proceso de desarrollo.

1.1.2 Factores que determinan la calidad de un producto de software.

Un factor de calidad de software es un requisito deseable que mejora la calidad de los programas de software. Son características que se buscan para maximizar el propio software y optimizar su calidad. Los factores que miden la calidad del software se centran en tres aspectos importantes de un producto software (Rubio, 2002):

- 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.

1.1.3 Aseguramiento de la Calidad y Pruebas de Software.

Cuando se refiere a la calidad de software es necesario tratar dos definiciones importantes: aseguramiento de la calidad y pruebas de software, aunque las pruebas de software son una parte del proceso de aseguramiento de calidad.

El aseguramiento de la calidad del software es el conjunto de actividades planificadas y sistemáticas necesarias para optar la confianza en que el producto satisfaga los requisitos dados de calidad. El aseguramiento pretende dar confianza o garantiza en qué el producto tiene calidad. (Lovelle, 1999)

“El aseguramiento de la calidad se construye dentro del proceso, en todas las actividades, involucrando a todos los participantes, utilizando medidas y criterios objetivos, permitiendo así detectar e identificar los defectos en forma temprana.” (Metodologías - RUP, 2007).

Las pruebas de software son el proceso de validación y comprobación que un programa de software debe cumplir en cuanto a los requisitos técnicos y comerciales que han guiado su diseño y desarrollo, de esta forma, se puede asegurar que funcione correctamente el producto final.

Las pruebas de software son una fase importante dentro de casi todos los modelos conocidos de ciclo de vida del software y aseguramiento de la calidad, se referirá entonces, a asegurar la calidad de los resultados de cada una de las fases del ciclo de vida del software (incluida la de pruebas) y con esto, asegurar la calidad del producto. (Pressman, 2002)

Una vez conocidas estas definiciones es factible mencionar algunas diferencias y relaciones entre ambos:

- Las Pruebas de Software utilizarán casos de pruebas para ser ejecutados; en cambio para el aseguramiento de la calidad de software se utilizará los estándares y procedimientos establecidos para cada una de las fases del ciclo de vida del software.
- Ambas permiten verificar y confirmar la calidad del producto final.
- Ambas definen un conjunto de actividades a realizar dentro del ciclo de vida del software para mejorar y asegurar la calidad del mismo.

1.1.4 Modelos de Calidad.

A partir del momento en que las propias organizaciones que trabajan en ámbitos educativos y de acción social sienten y perciben la necesidad de establecer mecanismos de mejora para asegurar la calidad. Se empiezan a establecer y definir procesos y principios que facilitan el cambio. Un cambio exigido por un conjunto de nuevas demandas y necesidades sociales y por un claro sentido de transparencia ante la sociedad a través de modelos de calidad.

Un modelo de calidad es un conjunto de prácticas vinculadas a los procesos de gestión y al desarrollo de proyectos. Estos modelos se planifican para alcanzar un impacto estratégico, cumpliendo con los objetivos fijados en lo referente a la calidad del producto o servicio. (Daily, 2008)

Según la investigación que se ha llevado a cabo los modelos de calidad precisan una diferenciación conceptual para evitar cualquier confusión existente en relación con los términos de calidad. Los modelos más utilizados son CMM, ISO/IEC TR 15504 e ISO 9000 - 2000.

A continuación se mencionan algunas ventajas y desventajas de cada modelo según Armando Silva Alarcón:

Capability Maturity Model (CMM). (ALARCÓN, 2004)

Es un marco evolutivo organizado en cinco niveles para lograr la mejora continua de procesos. CMM para el software es un conjunto ampliamente aceptado de directrices para el desarrollo de alto rendimiento en

las organizaciones. Determina que la calidad de una solicitud se relaciona directamente con la calidad del proceso utilizado para su desarrollo.

Ventajas

- Específico para el desarrollo y mantenimiento de software.
- Definido como un conjunto de áreas clave de procesos.
- Tiene un modelo de evaluación.
- Existen organizaciones evaluadoras.

Desventajas

- Es un modelo extranjero, no internacional.
- No es fácil de aplicar (pensado para organizaciones grandes).
- La mejora no está enfocada directamente a los objetivos de negocio.
- La evaluación es costosa y no tiene período de vigencia.
- Se está abandonando a favor de CMM-I.

ISO/IEC TR 15504. (ALARCÓN, 2004)

Define el modelo de referencia de procesos de software y de capacidades de procesos que constituyen la base para la evaluación de procesos de software. Se compone de 9 partes de las cuales la 2, 3 y 9 son normativas y las demás informativas.

Ventajas

- Específico para el desarrollo y mantenimiento de software.
- Fácil de entender (24 procesos, 16 páginas).
- Definido como un conjunto de procesos.
- Orientado a mejorar los procesos para contribuir a los objetivos del negocio.

Desventajas

- No es práctico ni fácil de aplicar.
- Tiene solamente lineamientos para un mecanismo de evaluación.
- Todavía no es norma internacional.

Norma ISO 9000-2000. (ALARCÓN, 2004)

Es una norma internacional destinada a evaluar la capacidad de la organización para cumplir los requisitos del cliente, los reglamentarios y los propios de la organización. Esta norma define el modelo para el aseguramiento de la calidad, como el conjunto normalizado o seleccionado de requisitos del sistema.

Ventajas

- Tiene un mecanismo de certificación bien establecido.
- Está disponible y es conocida.

Desventajas

- No es específica para la industria de software.
- No es fácil de entender.
- No está definida como un conjunto de procesos.
- No es fácil de aplicar.

El mundo de la industria de software se ve en la gran necesidad dado el incremento de las TIC y del desarrollo del software a nivel mundial de desplegar aplicaciones que sean cada vez más robustas y con mayor complejidad en su totalidad. Por ende la calidad de estos software tiene que ser la más alta posible, brindándole al cliente una plena confianza de que su software garantiza lo que tiene que hacer en el momento dado, con las condiciones dadas. Para lograr una mayor calidad en el software no sólo se centra en las aplicaciones de los modelos que se plantean anteriormente, así como otros que existen en la actualidad, también se debe centrar los esfuerzos para lograr la calidad del software en las pruebas, que son las que pueden ofrecer un producto final con la menor cantidad de errores posibles, para que el mismo llegue al usuario final adecuadamente.

1.2 Descripción del objeto de estudio

1.2.1 Metodologías de desarrollo de software.

La obtención de productos de software con calidad, implica la utilización de metodologías o procedimientos, estándares para el análisis, el diseño, la programación y pruebas del mismo que permitan uniformar la filosofía de trabajo. Su objetivo 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 Desarrollo, RUP[1] por sus siglas en inglés. Este 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. Los diagramas presentados en dicho documento fueron modelados con la herramienta case Visual Paradigm UML 3.0, usando el Lenguaje Unificado de Modelado 2.0 (UML 2.0). (Ronquillo, 2009)

¿Qué es RUP?

RUP es una metodología de desarrollo que sirve de guía para indicar paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando qué personas deben participar en relación de las actividades y qué papel deben jugar en ello. Además, detalla la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.

Es uno de los procesos más generales de los existentes actualmente, pues está pensado para adaptarse a cualquier proyecto. Captura varias de las mejores prácticas en el desarrollo moderno de software en una forma que es aplicable para un amplio rango de proyectos y organizaciones. Es una guía de cómo utilizar de manera efectiva UML, provee a cada miembro de un equipo un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas de desarrollo. Crea y mantiene *modelos,* en lugar de enfocarse en la producción de una gran cantidad de papeles de documentación. (Soto, 2007)

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)

[1] Rational Unified Process

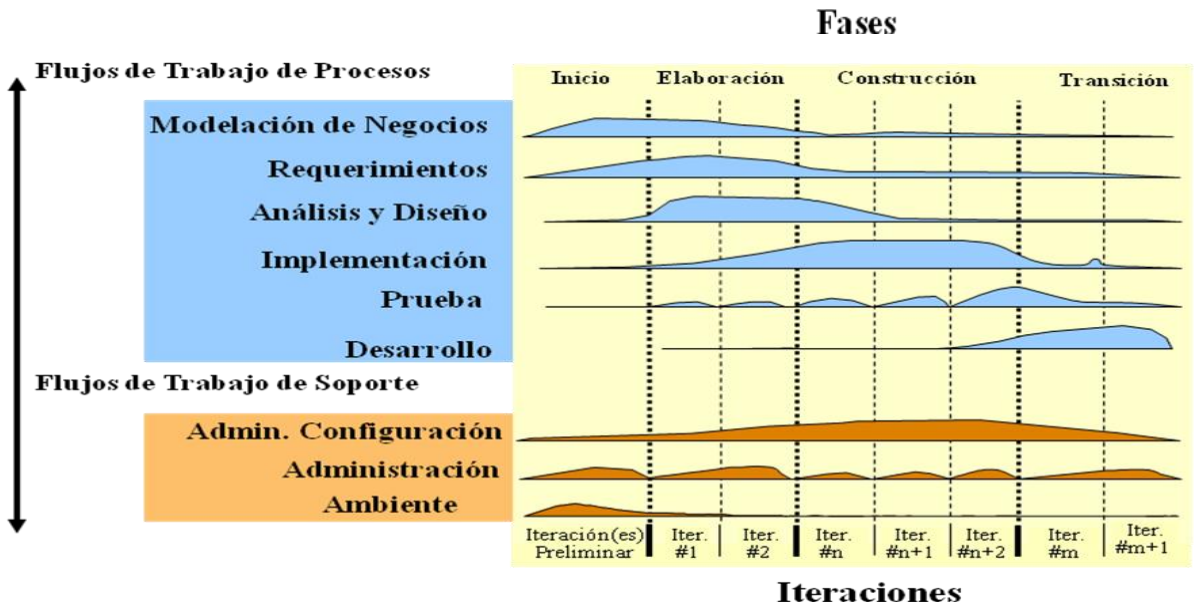

Figura 1 Flujos de trabajos de RUP

Se basa en casos de uso para describir lo que se espera del software y está muy orientado a la arquitectura del sistema, documentándose lo mejor posible basándose en UML como herramienta principal. (Molpeceres, 2003).

RUP provee a cada uno de los miembros del equipo las guías de proceso, mentores de herramientas, y plantillas, evidenciándose así las mejores prácticas: desarrollar iterativamente, administrar requisitos, arquitectura basada en componentes, modelar visualmente, verificar la calidad del software y controlar cambios. Además, es dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

1.2.2 Proceso de Pruebas de RUP.

Las pruebas son ejecutadas bajo condiciones o requisitos específicos, los resultados son observados y registrados, se hace una evaluación y rectificación de algunos aspectos del sistema o componente. Es un proceso iterativo que se lleva a cabo conjuntamente con la implementación y al finalizar esta actividad tienen lugar las pruebas del sistema. Este flujo de trabajo es el encargado de evaluar la calidad del producto que se está desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

Las pruebas no pueden asegurar la ausencia de defectos; sólo pueden demostrar que existen defectos en el software. (BRUEGGE, 2002)

Esta disciplina brinda soporte a las demás. Sus objetivos son (Pressman, 2002):

- Encontrar y documentar defectos en la calidad del software.
- Generalmente asesora sobre la calidad del software percibida.
- Provee la validación de los supuestos realizados en el diseño y especificación de requisitos por medio de demostraciones concretas.
- Verificar las funciones del producto de software según lo diseñado.
- Verificar que los requisitos tengan su apropiada implementación.

Las actividades de este flujo comienzan pronto en el proyecto con el plan de pruebas (el cual contiene información sobre los objetivos generales y específicos de las pruebas en el proyecto, así como las estrategias y recursos con que se dotará a esta tarea), o incluso antes con alguna evaluación durante la fase de inicio, y continuará durante todo el proyecto. (PRESSMAN, 2002)

El desarrollo del flujo de trabajo consistirá en planificar qué es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados, de forma que la información obtenida sirva para ir refinando el producto.

Los trabajadores que participan en este flujo de trabajo según plantea Roger Pressman en su última edición “Un Enfoque Práctico” del 2009 son:

Administrador de Pruebas: Es el responsable del éxito de la prueba, este rol es el defensor de la prueba y de la calidad, planifica y administra los recursos, resuelve los problemas que impidan las pruebas.

Analista de Pruebas: Es el responsable de identificar y definir las pruebas requeridas, monitorear el progreso de las mismas y el resultado en cada ciclo de pruebas, evaluando la calidad total experimentada como un resultado de las actividades de prueba. Este rol lleva la responsabilidad para representar apropiadamente las necesidades de los stakeholder que no tienen representación regular y directa en el proyecto.

Diseñador de Pruebas: Es el responsable de definir el método de prueba y asegurar su implementación exitosa. El rol incluye identificar técnicas apropiadas, herramientas e instrucciones para implementar las pruebas necesarias y encauzar los recursos correspondientes para las pruebas.

Probador: Conduce las pruebas necesarias y el registro del resultado de las mismas.

1.2.3 Artefactos del Flujo de Trabajo de Pruebas.

RUP plantea 7 artefactos fundamentales para el flujo de trabajo de prueba los cuales deben ser generados dentro de esta fase, pero específicamente en el presente trabajo se generan los artefactos de caso de prueba, plan de prueba y evaluación de las pruebas.

A continuación se nombran y se describen cada artefacto que genera RUP en el proceso de pruebas:

- **Modelo de prueba:** Describe fundamentalmente como se prueban los componentes en el modelo de implementación ejecutando pruebas de integración y de sistemas. Este artefacto puede describir también como han de ser probados aspectos específicos del sistema, por ejemplo si la interfaz de usuario es utilizable y consistente o si el manual de usuario del sistema cumple con su cometido.
- **Caso de prueba:** Especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse.
- **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.
- **Procedimiento de prueba:** Especifica cómo realizar uno o varios casos de prueba o partes de estos, por ejemplo un procedimiento de prueba puede ser una instrucción para un individuo sobre cómo ha de realizar un caso de prueba manualmente o puede ser una especificación de cómo interaccionar manualmente con una herramienta de automatización de pruebas para crear componentes ejecutables de prueba.
- **Plan de prueba:** Describe las estrategias, 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 de código necesario y el porcentaje de pruebas que deberían ejecutarse con un resultado específico.
- **Defecto:** Es una anomalía del sistema, como por ejemplo un síntoma de un fallo de software o un problema descubierto en una revisión. Un defecto puede ser utilizado para localizar cualquier cosa que los desarrolladores necesitan registrar como síntoma de un problema en el sistema que ellos necesitan controlar y resolver.
- **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.

1.2.4 Actividades fundamentales del flujo de trabajo de prueba.

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

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

1.2.5 Documentos del proceso de pruebas utilizados en el módulo Registro Petrolero.

En el proceso de pruebas a realizar al módulo Registro Petrolero se generan los siguientes artefactos:

El plan de prueba: Describe todos los métodos que se utilizarán para verificar que el software satisface la especificación del producto y las necesidades del cliente. Incluye los objetivos de calidad, necesidades de recursos, crono grama, asignaciones, métodos, entre otros.

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

El Informe de No Conformidades: Documento en el cual se archivan los incumplimientos de los requisitos definidos por el cliente. Define el procedimiento interno a seguir por los desarrolladores del proyecto para el tratamiento de los errores cometidos.

1.2.6 Elementos a tener en cuenta para que una prueba tenga éxito.

Estrategia de Prueba.

Describe el enfoque y los objetivos generales de las actividades de prueba. Incluye los niveles de prueba (unidad, integración, etc.) a ser diseccionados, el tipo de prueba a ser ejecutada (funcional, stress, etc.) y los casos de prueba diseñados para lograr los objetivos.

Define: Técnicas de pruebas (manual o automática) y herramientas a ser usadas. Criterios de éxitos y culminación de las pruebas. Consideraciones especiales relacionadas con los recursos necesarios para realizar las pruebas.

Para probar los sistemas Orientado a Objetos adecuadamente, se deben hacer tres cosas:

- Incluir técnicas de detección de errores aplicados a los modelos de análisis y diseño Orientado a Objetos.
- Cambiar la estrategia para las pruebas de unidad e integración. Comenzar por probar las clases y en las pruebas de integración ascendente y descendente se elimina.
- El diseño de casos de prueba debe tener en cuenta las características propias del software orientado a objetos.

Niveles de Prueba.

La Prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo.
Se distinguen los siguientes niveles de pruebas: Prueba de desarrollador, Prueba independiente, **Prueba de Unidad**, **Prueba de Integración** (Ascendente Basadas en hilo y Descendente Regresión basados en Uso), **Prueba de sistema** y Prueba de aceptación o **Validación**. (Pressman, 2002)

- Prueba de Unidad

En programación, una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las Pruebas de Integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión. Estas pruebas suelen realizarlas el propio personal de desarrollo, pero evitando que sea el propio programador del módulo.

- Prueba de Integración

Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez. Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos. Las pruebas de integración (algunas veces llamadas integración y testeo I & t) es la fase del testeo de software en la cual módulos individuales de software son combinados y testeados como un grupo. Son las pruebas posteriores a las pruebas unitarias y preceden el testeo de sistema.

- Prueba Integración Descendente.

  La Prueba de Integración Descendente es un planeamiento incremental a la construcción de la estructura de programas. Se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el módulo de control principal (programa principal).

- Prueba Integración Ascendente.

  La Prueba Integración Ascendente, como su nombre lo indica, empieza la construcción y la prueba con los módulos atómicos (es decir, módulos de los niveles más bajos de la estructura del programa).

➢ Prueba de Sistema

Está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadoras. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas.

Entre las pruebas del sistema se pueden considerar las siguientes:

- Prueba de Recuperación.

  La Prueba de Recuperación es una prueba que se le hace al sistema que fuerza el fallo del software de muchas formas y verifica que la recuperación se lleve a cabo apropiadamente.

- Prueba de Seguridad.

  La Prueba de Seguridad intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán de accesos impropios.

- Prueba de Resistencia.

  La Prueba de Resistencia ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales.

- Prueba de Rendimiento.

  La Prueba de Rendimiento está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

- Prueba de instalación

  Se centra en asegurar que el sistema software desarrollado se puede instalar en diferentes configuraciones hardware y software y bajo condiciones excepciones, por ejemplo con espacio de disco insuficiente o continuas interrupciones.

- Prueba de funcionalidad

  La prueba de funcionalidad fija su atención en la validación de las funciones, métodos, servicios, caso de uso.

Se usan para probar que el sistema funciona correctamente en conjunto. Cada prueba de sistema prueba principalmente combinaciones de casos de usos instanciados bajo condiciones diferentes. Estas condiciones incluyen diferentes configuraciones hardware (procesadores, memoria principal, discos duros, etc.), diferentes niveles de carga del sistema, diferentes números de actores y diferentes tamaños de la base de datos.

- Pruebas de Validación

Verifica que el software producido cumple con las especificaciones requeridas por el cliente. Es normalmente una parte del proceso de pruebas de un proyecto, utiliza técnicas tales como evaluaciones, inspecciones y tutoriales.
Evalúa el sistema para determinar si satisface los requisitos iniciales. La pregunta a realizarse es: ¿Es esto lo que el cliente desea?

Caso de Prueba.

Conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada. Siempre es ejecutada como una unidad, desde el comienzo hasta el final. (Pressman, 2009)

Debe verificar:

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

Id del escenario	Escenario	Titulo	Cuerpo		Respuesta del Sistema	Resultado de la Prueba
EC 1	Escenario 1: Actualizar Reseña Histórica	V Antecedentes de la investigación	V Descripción		El sistema debe mostrar un mensaje informándole al administrador que ya han sido actualizados los temas de asesoría	

Tabla 1 Ejemplo de Caso de Prueba

1.2.7 Diseño de Casos de Prueba.

El diseño de casos de prueba es una parte de las pruebas de componentes y sistemas en la que se diseñan los casos de prueba (entradas y salidas esperadas) para probar el sistema. El objetivo del proceso de diseño de casos de prueba es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requerimientos. Para diseñar un caso de prueba, se selecciona una característica del sistema o componente que se está probando.

El diseño de pruebas para el software o para otros productos de ingeniería puede requerir tanto esfuerzo como el propio diseño inicial del producto. Sin embargo, los ingenieros de software, por razones que ya han tratado, a menudo tratan las pruebas como algo sin importancia, desarrollando casos de prueba que perezcan adecuados, pero que tienen pocas garantías de ser completos. Recordando el objetivo de las pruebas, debemos diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo posible. (PRESSMAN, 2004)

1.2.8 Tipos de Prueba.

Las técnicas o tipos de pruebas son un conjunto integrado de acciones que combinadas de forma general, miden, evalúan, y verifican disimiles parámetros de calidad de software, para así lograr el objetivo trazado con la realización de las pruebas.

Cada tipo de prueba tiene un objetivo específico y una técnica que lo soporte para cada nivel de esta. El siguiente esquema muestra los tipos de pruebas basados en dimensiones de calidad. (JACOBSON, y otros, 2002).

Dimensión de Calidad	Tipos de prueba
Funcionalidad	Función, Seguridad, Volumen
Usabilidad	Usabilidad
Fiabilidad	Integridad, Estructura, Stress
Rendimiento	Contención, Carga, Profile, benchmark
Soportabilidad	Configuración, Instalación

Tabla 2 Tipos de Prueba

A continuación se identifican algunas de las técnicas a utilizar en el proceso de pruebas del módulo Registro Petrolero:

- **Pruebas de Funcionalidad**

Estas pruebas se aplicarán para comprobar que el sistema fue implementado y funcione correctamente de acuerdo con los requisitos establecidos por el cliente. El objetivo principal es de asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de los resultados, teniendo como metas verificar el procesamiento, recuperación e implementación adecuada de las reglas del negocio y la apropiada aceptación de los datos. Con el método de Caja Negra se ejecuta cada caso de uso, flujo de caso de uso o función, usando datos válidos e inválidos para verificar lo siguiente (Pressman, 2002):

- Que se aplique apropiadamente cada regla de negocio.
- Que los resultados esperados ocurran cuando se usen datos válidos.
- Que sean desplegados los mensajes apropiados de error y precaución cuando se usan datos inválidos.

- **Pruebas de Seguridad**

Estas pruebas se realizarán para comprobar la seguridad que brinda el sistema para proteger la información que almacena; es decir que sólo puedan acceder a él los usuarios registrados y que estos a su vez tengan acceso a la información relacionada con su rol. (Pressman, 2002)

Las áreas que abarca, seguridad del sistema incluyendo acceso a datos o funciones de negocios, seguridad del sistema incluyendo ingresos y accesos remotos al sistema. Garantizando que los usuarios restringidos a funciones específicas o su acceso están limitado únicamente a los datos que está autorizado a acceder, y que sólo aquellos usuarios autorizados a acceder al sistema son capaces de ejecutar las funciones del sistema.

Las técnicas que se utilizarán son las siguientes: identificar cada tipo de usuario, las funciones y datos a los que se debe autorizar, crear pruebas para cada tipo de usuario y verificar cada permiso creando transacciones específicas para cada tipo de usuario, además de modificar cada tipo de usuario y volver a ejecutar las pruebas.

- **Pruebas de Estructura**

La prueba de estructura se implementa y se ejecuta para verificar que todos los enlaces (estáticos o activos) están conectados correctamente. Se centran en la evaluación de la adherencia del destino de la prueba a su diseño y formación. Normalmente, se realiza en aplicaciones habilitadas para web y garantiza que se muestre el contenido adecuado según el enlace. (Pressman, 2002)

La clave es que todos los resultados de las decisiones deben ejercitarse de forma independiente durante la prueba y que el número de pruebas necesario para un módulo de software es igual a la complejidad ciclomática de dicho módulo. Estas pruebas incluyen: (Pressman, 2002)

- Verificación de que se muestra el contenido correcto (texto, gráficos, etc.) de cada enlace. Se utilizan diferentes tipos de enlaces para hacer referencia al contenido de destino de las aplicaciones basadas en web, como los marcadores, hiperenlaces a otro contenido de destino (en el mismo sitio web o en uno diferente) o zonas activas. Deben comprobarse todos los enlaces para garantizar que se muestra el contenido de destino correcto a los usuarios.

- Comprobación de que no hay enlaces rotos. Son los enlaces cuyo contenido de destino no se puede encontrar. Estos pueden estar rotos por muchos motivos, incluidos el desplazamiento, la eliminación o el cambio de nombre de los archivos de contenido de destino. Los enlaces también

pueden estar rotos debido a un uso incorrecto de la sintaxis, incluidos los dos puntos, las barras inclinadas o las letras que falten.

- Verificación de que no hay contenido huérfano. Son los archivos que no tienen enlaces "de entrada" en el sitio web; es decir, no se puede acceder a su contenido ni se puede presentar. Debe investigarse con cuidado el contenido huérfano para determinar la causa:

¿Es huérfano porque realmente ya no es necesario?
¿Es huérfano debido a un enlace roto?
¿O se accede a él a través de un enlace externo al sitio web actual?

Una vez que se haya determinado la causa, debe llevarse a cabo la acción adecuada, como eliminar el archivo de contenido, reparar el enlace roto o ignorar el huérfano, respectivamente. (Pressman, 2002)

1.2.9 Método de Prueba de Caja Blanca.

En las pruebas de Caja Blanca se comprueban los caminos lógicos del software proponiendo casos de pruebas que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado. (Sobre el código). (Pressman, 2009).

Consiste en realizar pruebas para verificar que líneas específicas de código funcionan tal como está definido. También se le conoce como prueba de caja-transparente o prueba de caja de cristal. (Ambler, 2004).

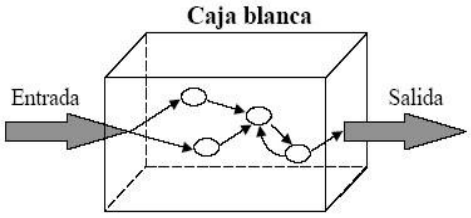

Figura 2 Prueba de Caja Blanca

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 pruebas que garanticen que:

- Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Se ejerciten las estructuras internas de datos para asegurar su validez.

Algunas técnicas empleadas en las pruebas de caja blanca son los siguientes:

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.

Los pasos que se siguen para aplicar esta técnica son:

- A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo.
- Se determina un conjunto básico de caminos independientes.
- Se preparan los casos de pruebas que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. Es el acto de asegurar que todos los caminos lógicos en el código se ejercitan al menos una vez.

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:

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.

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).

En la presente investigación del método de Caja Blanca se hará uso de la técnica del camino básico.

1.2.10 Método de Prueba de caja Negra.

Las pruebas de Caja Negra se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código).

La prueba verifica que el ítem que se está probando, cuando se dan las entradas apropiadas produce los resultados esperados. (Ambler, 2004).

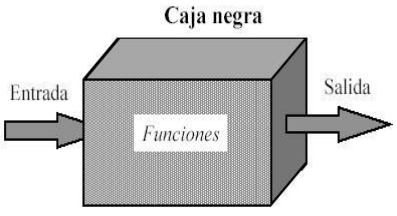

Figura 3 Prueba de Caja Negra

Se centran principalmente en los requisitos funcionales del software. Permiten encontrar: Funciones incorrectas o ausentes, Errores de interfaz, Errores en estructuras de datos o en accesos a las bases de datos externas, Errores de rendimiento, Errores de inicialización y terminación.

Algunas técnicas empleadas en las pruebas de caja negra son los siguientes:

Partición Equivalente

La partición equivalente es 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 estos requerían 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 de esa forma el número total de casos de prueba que hay que desarrollar.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada.

Quién va a representar un conjunto de estados válidos o no válidos para condiciones de entrada se denomina clase de equivalencia. Una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

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

Grafos de Causa y Efectos

Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Análisis de valores límite

Los errores tienden a darse con más frecuencia en los límites del campo de entrada que en el “centro”. Para ello se ha desarrollado una técnica de análisis de valores de límites (AVL) para ser aplicadas a las pruebas. Su análisis nos lleva a una elección de casos de prueba que ejerciten los valores límite. El mismo es una técnica de diseño de casos de prueba que complementa a la partición equivalente. En vez 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. Así mismo en lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida.

Las directrices de AVL son similares en muchos aspectos a las que proporciona la partición equivalente:

- Si una condición de entrada especifica un rango delimitado por los valores a y b, se deben diseñar casos de prueba para los valores a y b, y para los valores justos por debajo y justo por encima de a y b, respectivamente.
- Si una condición de entrada específica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximo y mínimo. También se deben probar los valores justo por encima y justo por debajo del máximo y del mínimo.
- Aplicar las directrices 1 y 2 a las condiciones de salida. Por ejemplo, supongamos que se requiere una tabla de (temperatura/presión) como salida de un programa de análisis de ingeniería. Se deben diseñar casos de prueba que creen un informe de salida que produzca el máximo (y el mínimo) numero permitido de entradas en la tabla.
- Si las estructuras de datos internas tienen límites preestablecidos (por ejemplo, una matriz que tenga un límite definido de 100 entradas), hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.

La gran parte de los ingenieros del software lleva a cabo de forma intuitiva alguna forma de AVL. Con la aplicación de las directrices que se acaban de exponer, la prueba de límites será más completa y, por tanto, tendrá una mayor probabilidad de detectar errores.

Prueba de comparación

Hay situaciones en las que la fiabilidad del software es algo absolutamente crítico. En ese tipo de aplicaciones a menudo se utiliza hardware y software redundante y de esta forma se puede minimizar la posibilidad de error. Cuando se desarrolla un software redundante varios equipos de ingeniería del software separados desarrollan versiones independientes de una aplicación, usando las mismas especificaciones. Posteriormente se procede a probar todas las versiones con los mismos datos de

prueba, para verificar si todas proporcionan una salida idéntica, luego se ejecutan todas al mismo tiempo y se comparan los resultados en tiempo real.

Prueba de tabla ortogonal

La prueba de la tabla ortogonal es aplicable a problemas en que el dominio de entrada es relativamente pequeño pero demasiado grande para posibilitar pruebas exhaustivas. Este método es particularmente útil para encontrar errores asociados con fallos localizados, una categoría de error asociada con defectos de la lógica dentro de un componente software.

La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca.

En la presente investigación del método de Caja Negra se hará uso de la técnica de participación equivalente.

1.3 Soluciones existentes y herramientas para automatizar las pruebas.

Durante el estudio de la investigación se verificó que no existen soluciones de pruebas realizadas al módulo Registro Petrolero. Con el diseño y aplicación de las pruebas se obtendrá en su primera versión los defectos encontrados en dicho módulo, estos serán documentados y eliminados, comprobando que el módulo realice sus funciones específicas requeridas por el cliente.

Existen varias herramientas de automatización de las diferentes técnicas de pruebas. Entre ellas se pueden citar:

VTest: Es una herramienta automática para pruebas funcionales y de regresión para aplicaciones web. Incorpora capacidades de registro, verificación, reproducción y reporte. No requiere conocimientos de programación. Para aquellos usuarios que desean escribir programas, éste usa JavaScript como la lengua de programación. Soporta tanto Microsoft Internet Explorer como Mozilla Firefox. (Hower, 2010)

AdventNet QEngine: Es una herramienta para automatizar pruebas de funcionalidad, rendimiento, carga, tráfico de datos, servicios web, control del rendimiento de servidores y pruebas de regresión, para ejecutar pruebas desde la línea de comandos para todo tipo de pruebas. Funciona tanto en plataformas Windows

como en Linux. Permite grabar y reproducir los exploradores Microsoft Internet Explorer, Firefox y Mozilla. (Hower, 2010)

AdventNet QEngine WebTest: Posee una amplia gama de herramientas independientes de la plataforma, para pruebas de funcionalidad y de carga de la Red en aplicaciones web desarrolladas usando HTML, JSP, ASP, .NET, PHP, JavaScript/VBScript, e-commerce, etc. La herramienta de pruebas de Red ha sido desarrollada en Java, lo que facilita su portabilidad, soporte para múltiples plataformas (Windows, Linux, y Solaris) y para múltiples navegadores (Firefox, IE y Mozilla). (Hower, 2010)

Selenium IDE: Es una herramienta para realización de pruebas de aplicaciones web, está orientado a la ejecución de pruebas funcionales a nivel de usuario directamente desde el navegador. Esta herramienta es muy útil para el desarrollo web donde se tienen que realizar montones de pruebas cada vez que se saca una versión nueva o se realizan modificaciones en un portal. Selenium automatiza el proceso de pruebas y permite ejecutar un conjunto de pruebas completo si es necesario o pruebas particulares. (Hower, 2010)

Las técnicas de pruebas seleccionadas no requieren una herramienta que las automatice, debido a que la complejidad de su ejecución es baja, y se pueden llevar a cabo manualmente. Por ende no se aplicara ninguna de estas herramientas para la automatización de las pruebas en el módulo Registro Petrolero.

Conclusiones parciales

A partir del análisis de los aspectos esenciales de este capítulo se evidencia la importancia que tienen las pruebas en la calidad del software, se caracteriza la metodología RUP como guía a seguir en el proyecto SGDG, se definen las características de niveles, técnicas, métodos, tipos, procedimientos, planes de pruebas y herramientas, decidiéndose aplicar los métodos y técnicas de pruebas de caja blanca “camino básico” y el método de prueba de caja negra “partición equivalente” que se emplean para la realización de las mismas, diseñar los casos de pruebas necesarios y probar las funcionalidades del módulo Registro Petrolero permitiendo documentar y corregir cualquier error.

Capítulo 2: Diseño y aplicación de las Pruebas.

Introducción

El diseño y aplicación de las pruebas al módulo Registro Petrolero precisará un plan de pruebas como base para todo el proceso que se llevará a cabo, así como los requisitos funcionales, casos de uso y casos de pruebas diseñados, especificando cada uno de los elementos de la prueba definidas.

2.1 Características a probar

Las características a evaluar en el módulo Registro Petrolero con la realización de las pruebas son: si la aplicación cumple con las funcionalidades requeridas desde el comienzo. Si no tiene errores a la hora de ejecutarlo, si es comprensible para el usuario, si se puede usar correctamente con facilidad y si cuenta con la documentación detallada que correspondan con el módulo.

2.2 Plan de Pruebas

Señalar el enfoque, los recursos y el esquema de actividades de pruebas, así como los elementos a probar, las características, las actividades de prueba, el personal responsable y los riesgos asociados. Tiene como objetivo el desarrollo de una hoja de ruta para la realización de las pruebas, este debe ser documentado y aprobado por los participantes del mismo. En el plan de prueba definido para el módulo Registro Petrolero se identifican los elementos que serán aprobados, los recursos necesarios para hacer las pruebas, así como la estrategia de pruebas que se llevará a cabo para lograr un buen diseño de casos de pruebas que permitan encontrar la mayor cantidad posible de defectos.

2.2.1 Listas de chequeo

La lista de chequeo es un instrumento de medición y evaluación que consiste básicamente en un formulario de preguntas referentes al atributo de calidad que se está probando y de las características del documento en el caso de la documentación. Cada pregunta tiene asociada una evaluación en una escala que da una medida del grado de cumplimiento y disponibilidad de la propiedad evaluada, de esta manera, se determina la evaluación del elemento probado.

En materia de documentación se utilizará una lista de chequeo por cada documento revisado, dígase Manual de usuario y Especificación de requisitos.

Se tendrán en cuenta aspectos significativos como:

Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.
Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.
Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.

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 Manual de usuario:

¿Se especifica detalladamente cada interfaz del flujo de eventos a probar?
¿Se explicitan todos los campos y funcionalidades de éstos por interfaz?
¿Corresponde la verdadera interfaz de usuario con la señalada en el documento?
¿Se han identificado errores ortográficos?
¿Se entiende claramente lo que se ha especificado en el documento?

En el documento de Especificación de requisitos:

¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?
¿Debería especificarse algún requisito con más detalle?
¿Todos los requisitos identificados se centran en lo que el sistema debe hacer y no como el sistema debe hacerlo?
¿Se han identificado los requisitos de software y de hardware?
¿Los requisitos de soporte y usabilidad se han identificado?
¿Se puede verificar cada requisito? (Un requisito se dice que es verificable si existe algún proceso no excesivamente costoso por el cual una persona o una máquina pueda chequear que el software satisface dicho requisito, ejemplo la especificación del caso de uso).
¿Se han enumerado los requisitos incluso los que se derivan de otros requisitos?
¿Se han identificado errores ortográficos?
¿Se entiende claramente lo que se ha especificado en el documento?

Para probar la aplicación las listas de chequeo verifican el cumplimiento de los atributos de calidad enmarcándose en la verificación de la confiabilidad, seguridad, portabilidad, usabilidad y eficiencia.

Para este caso las preguntas que se tuvieron en cuenta por cada atributo se muestran a continuación.

Usabilidad

¿Proporciona el sistema un soporte apropiado a los usuarios más novatos?

¿Resulta fácil instalar el software?

¿Se entienden la interfaz y su contenido?

¿Resulta fácil entender el resultado de una acción?

¿Asiste el sistema al usuario de forma efectiva proporcionando el modo más efectivo de hacer las tareas en caso de que no haya una única forma de hacerlas?

¿Es fácil de utilizar el sistema en la realización de tareas?

¿Está diseñada la interfaz para facilitar la realización eficiente de las tareas de la mejor forma posible?

¿Actúa el sistema como corrector de errores?

¿Actúa el sistema en la prevención de errores?

¿Actúa el sistema en la información de los errores?

¿Se permite la utilización del ratón o el teclado?

¿Permite al usuario interrumpir su tarea y continuar más tarde?

¿Permite una cómoda navegación dentro del producto y una fácil salida de éste?

¿Permite distintos niveles de uso del producto para usuarios con distintos niveles de experiencia?

¿Proporciona funciones deshacer, rehacer?

¿Se presenta al usuario la información que sólo necesita?

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)?

2.2.2 Recursos para las Pruebas

Para la realización de las pruebas se hace necesario contar con algunos recursos tangibles dentro de los cuales se encuentran:

Requisitos de Software:

- Las computadoras que utilizarán el software deben tener instalado:
  - Windows 2000 NT, Windows XP Profesional o 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.

Requisitos de Hardware:

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

En este proceso de pruebas se tienen en cuenta los requisitos, que dan conocimiento la funcionalidad del sistema, pues constituyen condiciones o capacidades que éste debe cumplir y forman parte de la documentación del proyecto. (Ronquillo, 2009)

2.2.3 Requisitos Funcionales y Casos de Usos

A continuación se especifican una serie de casos de usos y requisitos que enmarcan la funcionalidad del módulo Registro Petrolero, estos se describen a continuación:

Módulo Registro Petrolero

Caso de Uso 1: Gestionar Libros de Registros (Crítico): El caso de uso comienza cuando el Especialista de Registro, Control y Asesoría Legal accede a la opción de Insertar, Eliminar, o Modificar Libro de Registro del menú de operaciones. El caso de uso termina cuando el sistema devuelve la información solicitada, y el usuario accede a otra opción o simplemente sale de la aplicación.

Caso de Uso 2: Gestionar Solicitudes de Calificación (Crítico): El caso de uso comienza cuando el Especialista de Registro, Control y Asesoría Legal accede a la opción de Insertar, Eliminar, Modificar y Mostrar Cantidad del menú de operaciones. El caso de uso termina el sistema realiza la operación elegida, y el Especialista accede a otra opción o simplemente sale de la aplicación.

Caso de Uso 3: Gestionar Solicitudes de Permiso de explotación (Crítico): El caso de uso comienza cuando el Especialista de Registro, Control y Asesoría Legal accede a la opción de Solicitudes de Permiso de Perforación en el menú “Gestión”, para realizar las acciones de buscar, adicionar, modificar, eliminar y ver detalles de una solicitud. El caso de uso termina cuando el sistema realiza la operación elegida, y el Especialista accede a otra opción o simplemente sale de la aplicación.

Caso de Uso 4: Gestionar Documentos (Crítico): El caso de uso comienza cuando el Especialista de Registro, Control y Asesoría Legal accede a la opción de Insertar, Eliminar, o Modificar Documentos del menú de operaciones. El caso de uso termina cuando el sistema devuelve la información solicitada, y el usuario accede a otra opción o simplemente sale de la aplicación.

Caso de Uso 5: Mostrar Detalles: El caso de uso comienza cuando el usuario, que puede ser el Especialista de Control, Registro y Asesoría Legal (Especialista RCAL) u otro especialista del Departamento desee buscar todos los Detalles de una compañía.

Caso de Uso 6: Mostrar Datos de los procesos: El caso de uso comienza cuando el usuario, que puede ser el Especialista de Control Registro y Asesoría Legal (Especialista RCAL) u otro especialista del Departamento desee buscar los datos referentes a los cuatro procesos, siendo en un determinado período o no.

Caso de Uso 7: Mostrar Datos del Libro: El caso de uso comienza cuando el usuario, que puede ser el especialista de control, registro y asesoría legal (Especialista RCAL) u otro especialista del Departamento que desee buscar datos del libro seleccionado.

Caso de Uso 8: Generar Documentos (Crítico): EL sistema debe permitir al usuario seleccionar el documento que desea generar y la solicitud a la que pertenece. Como resultado de esta acción se mostrará el documento en formato PDF, dando la opción de imprimirlo.

Caso de Uso 9: Insertar Prórroga (Crítico): Con este requisito se podrá registrar en las notas marginales de las solicitudes de Permisos de Perforación, la solicitud de prórroga que realice determinado cliente de la ONRM para seguir trabajando en el territorio del país, así como también se insertará en el sistema la fecha hasta la que estará vigente la prórroga.

2.4 Diseño de Casos de Prueba de Caja Negra. Técnica, Partición de Equivalencia

Con las pruebas de Caja Negra se comprueba la funcionalidad del sistema, para el desarrollo de las mismas se aplica la técnica de partición de equivalencia que resulta efectiva a la hora de comprobar la validez de cada entrada.

El diseño de casos de prueba según esta técnica consta de dos pasos:

- Identificar las clases de equivalencia.
- Identificar los casos de prueba.

Identificar las clases de Equivalencia

Se definen dos tipos de clases de equivalencia, las clases de equivalencia válidas, que representan entradas válidas al programa, y las clases de equivalencia no válidas, que representan valores de entrada erróneos. Estas clases se pueden representar en una tabla.

Condición Externa	Clases de Equivalencias Válidas	Clases de Equivalencias No Válidas

Tabla 3 Tabla para la identificación de clases de equivalencia

Identificar los casos de prueba

Para crear los casos de prueba a partir de las clases de equivalencia se han de seguir los siguientes pasos:

- Asignar a cada clase de equivalencia un número único.
- Hasta que todas las clases de equivalencia hayan sido cubiertas por los casos de prueba, se tratará de escribir un caso que cubra tantas clases válidas no incorporadas como sea posible.
- Hasta que todas las clases de equivalencia no válidas hayan sido cubiertas por casos de prueba, escribir un caso para cubrir una única clase no válida no cubierta.

La razón de cubrir con casos individuales las clases no válidas es que ciertos controles de entrada pueden enmascarar o invalidar otros controles similares.

2.4.1 Diseño de Prueba - CN del Caso de Uso: Gestionar Solicitud de Calificación

Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. Este diseño se elabora previamente a realizar las pruebas exploratorias, o de interfaz como también pueden llamarse, se parte de la descripción de los casos de usos del sistema, como apoyo para las revisiones. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión, además quedan plasmadas las revisiones realizadas al caso de prueba; así como un registro de todo aquello que no satisface y corresponde a la calidad del software.

Existen casos de pruebas para los diferentes métodos: Caja Negra y Caja Blanca. En el proceso de prueba en cuestión se hace uso de los dos casos de prueba. 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.

El siguiente es un ejemplo donde se detalla el caso de uso Gestionar Solicitudes de Calificación del módulo Registro Petrolero que forma parte del Sistema de Gestión de Datos Geológicos.

Nombre del CU: Gestionar Solicitudes de Calificación.

Descripción General: El caso de uso comienza cuando el administrador accede a la opción de Adicionar, Eliminar, Modificar, Buscar o Mostrar detalles de una Solicitud de Calificación. El caso de uso termina cuando el sistema realiza la operación elegida, y el administrador accede a otra opción o simplemente sale de la aplicación.

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

Secciones a probar en el Caso de Uso:

- Adicionar Solicitudes de Calificación.
- Buscar Solicitudes de Calificación.

- Eliminar Solicitudes de Calificación.
- Modificar Solicitudes de Calificación.
- Mostrar Solicitudes de Calificación.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
*SC1:*Adicionar Solicitudes de Calificación	*EC 1.1:* El usuario selecciona "Adicionar".	El sistema muestra el formulario con los campos a llenar correspondiente a esta opción: Solicitante, Fax, Teléfono, Correo electrónico, Nacionalidad, Fecha de Solicitud. Así como un botón para adicionar.
	EC 1.2: El usuario introduce datos en los campos correctamente.	El sistema crea una nueva Solicitud de Calificación.
	EC 1.3: El usuario no introduce datos en alguno de los campos o en caso de hacerlo incorrectamente.	El sistema marca en rojo los campos que no han sido introducidos o que presentan datos incorrectos.
	EC 1.4: El usuario introduce datos en los campos correctamente pero ya existe la Solicitud de Calificación.	El sistema muestra un mensaje informando al usuario que ya esa Solicitud existe.
SC2 : Buscar Solicitudes de Calificación	*EC 2.1:* El usuario selecciona "Buscar".	El sistema muestra en el panel el listado de todas las Solicitudes con los datos: Identificador, Solicitante, Fecha de Solicitud, Estado, Nacionalidad, Acciones.
	EC 2.2: El usuario encuentra la Solicitud deseada en dicho listado.	Mantiene dicho listado.
	EC 2.3: Introduce el texto por el cual desea encontrar una Solicitud.	El sistema muestra en el panel el listado de Solicitudes con los datos: Identificador, Solicitante, Fecha de Solicitud, Estado, Nacionalidad,
SC3 : Eliminar Solicitudes de Calificación	*EC 3.1*: El usuario selecciona "Buscar", busca la Solicitud deseada y selecciona en acciones la opción eliminar Solicitud.	El sistema muestra un mensaje de confirmación "¿Está seguro que desea eliminar la Solicitud?".
	EC 3.2: El usuario acepta el mensaje de confirmación.	El sistema muestra un mensaje de confirmación "Solicitud Eliminada Satisfactoriamente" y muestra en "Buscar" el listado actualizado de las

	EC 3.3: El usuario cancela la acción.	El sistema cierra la ventana emergente y permanece en el mismo lugar.
SC4: Modificar Solicitudes de Calificación	*EC 4.1:* El usuario selecciona "Buscar", busca la Solicitud deseada y selecciona en acciones la opción modificar Solicitud.	El sistema muestra el panel con un formulario que contiene los datos: del Solicitante y da la Solicitud para que el usuario modifique el que desea.
	EC 4.2: El usuario modifica datos en los campos correctamente.	El sistema actualiza los datos de la Solicitud y actualiza el listado en el "Buscar".
SC5 : Mostrar Detalles de las Solicitudes de Calificación	*EC 5.1*: El usuario selecciona "Buscar", busca la Solicitud deseada y selecciona en acciones la opción Mostrar Detalles de las Solicitudes de Calificación.	El sistema muestra un panel con un formulario que contiene los datos: Solicitante, Fecha de Solicitud, Estado, Nacionalidad, Categoría, Tomo, Folio, Asiento, Fecha de Inscripción, Fecha de Certifico, Fecha Dictamen y Observaciones.
	EC 5.2: El usuario Observa los Detalles de la Solicitud de Calificación y le da al botón de cerrar.	El sistema cierra el panel y permanece en el mismo lugar.

Tabla 4 Diseño de Caso de Prueba de Caja Negra del Caso de Uso: Gestionar Solicitudes de Calificación

A partir de esta descripción se detallan las variables que se encuentran en toda la interfaz asociadas al caso de uso que se le estaba diseñando el caso de prueba. Esta descripción está compuesta por campos tales como:

- No: Se enumera todos los campos o variable, descrito en el caso de uso.
- Nombre de campo: Se especifica el nombre del campo de entrada.
- Clasificación: Se especifica la clasificación según el componente de diseño utilizado [ejemplo: campo de texto, lista desplegable o Lista de selección, entre otros].
- Puede ser nulo: Se especifica si el campo puede ser nulo o no, para ello solo se puso Sí o No.
- Descripción: Se describen brevemente los datos que debían introducirse.

En la siguiente tabla se muestra un ejemplo de la descripción de variables del caso de uso Gestionar Solicitudes de Calificaciones del módulo Registro Petrolero que forma parte del Sistema de Gestión de Datos Geológicos.

Descripción de Variables:

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*Solicitante*	*Texto*	*No*	*Se debe introducir en el campo de texto el nombre del solicitante*
2	Fax	Texto	Si	*Se debe introducir en el campo de texto el número de Fax*
3	Teléfono	Texto	Si	*Se debe introducir en el campo de texto el número de Teléfono.*
4	Correo Electrónico	Texto	Si	*Se debe introducir en el campo de texto el correo electrónico.*
5	Nacionalidad	Lista Desplegable	No	*Se debe desplegar el campo y escoger la nacionalidad.*
6	Fecha de Solicitud	Selección	No	*Se debe seleccionar la fecha.*
7	Buscar	Texto	No	*Se debe escribir lo que desea buscar.*
8	Eliminar	Selección	No	*Se debe seleccionar la Solicitud a eliminar.*
9	Mostrar	Selección	No	*Se debe seleccionar la Solicitud que desea mostrar sus datos.*

Tabla 5 Descripción de Variable del Caso de Uso Gestionar Solicitud de Calificación

Esta descripción posibilita que se ejecute una matriz de datos, donde se evalúa y se prueba la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estará probando. Utilizando un juego de datos válidos e inválidos se identificará el empleo de la técnica de partición de equivalencia.

Esta matriz de datos contiene los siguientes aspectos:

- Escenario: Se especifica el nombre del escenario.
- Variables [1, 2,…, n]: Se especifica el nombre de la variable, o el número según la tabla de descripción de variables y en su celda correspondiente se indicó el valor del dato [V (Válido), I (Inválido), N/A (No Aplica)].

- Respuesta del sistema: Se escribe el resultado que se esperaba al realizar la prueba.
- Resultado de la prueba: Se escribe el resultado que se obtuvo al realizar la prueba. (Satisfactorio o No Satisfactorio).
- Flujo central: Se expone el flujo central del caso de uso al que se le estaba diseñando el caso de prueba.

A continuación, un ejemplo de la matriz de datos del caso de uso Gestionar Solicitudes de Calificación del módulo Registro Petrolero que forma parte del Sistema de Gestión de Datos Geológicos.

Matriz de Datos

SC 1 Adicionar Solicitudes de Calificación

Escenario	V 1 Sol.	V 2 Fax	V 3 Teléf.	V 4 Correo	V 5 Nac.	V 6 F.de S.	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
El usuario selecciona "Adicionar"	*V*	***NA***	***NA***	***NA***	*V*	*V*	El Sistema muestra los campos a llenar.	*Satisfactorio*	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Calificacion de Solicitudes. 4- Adicionar
El usuario introduce datos en los campos correctamente.	*V*	*NA*	*NA*	*NA*	*V*	*V*	El sistema registra los datos.	*Satisfactorio*	
El usuario no introduce datos en alguno de los campos o en caso de hacerlo incorrectamente.	*I*	*NA*	*NA*	*NA*	*I*	*V*	El sistema muestra el campo de los datos vacios en rojo, y los no válidos colapsan el sistema.	*No Satisfactorio*	
El usuario introduce datos en los campos correctamente pero ya existe la Solicitud de Calificación.	*I*	*NA*	*NA*	*NA*	*I*	*V*	El sistema crea otra solicitud de Calificación.	*Satisfactorio*	

Tabla 6 Matriz de Datos. Adicionar Solicitud de Calificación

SC 2 Buscar Solicitudes de Calificación

Escenario	V 1 Sol.	V 2 Fax	V 3 Teléf.	V 4 Correo	V 5 Nac.	V 6 F.de S.	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
El usuario selecciona "Buscar"	V	NA	NA	NA	V	V	El Sistema Muestra una lista de Datos	Satisfactorio	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Calificacion de Solicitudes. 4- Buscar
El usuario encuentra la Solicitud deseada en	V	NA	NA	NA	V	V	Mantiene dicho Listado.	Satisfactorio	
Introduce en el campo texto datos por el cual desea encontrar una Solicitud.	V	NA	NA	NA	V	V	El sistema muestra la solicitud buscada.	Satisfactorio	
Introduce en el campo de texto datos Incorrectos.	I	NA	NA	NA	I	V	El sistema muestra; No se encontraron los elementos.	Satisfactorio	

Tabla 7 Matriz de Datos. Buscar Solicitud de Calificación

SC 3 Eliminar Solicitudes de Calificación

Escenario	V 1 Sol.	V 2 Fax	V 3 Teléf.	V 4 Correo	V 5 Nac.	V 6 F de S.	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
El usuario selecciona "Buscar", busca la Solicitud deseada y selecciona en acciones la opción eliminar Solicitud.	NA	NA	NA	NA	NA	NA	El sistema muestra un mensaje de confirmación "¿Está seguro que desea eliminar la Solicitud.	Satisfactorio	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Calificacion de Solicitudes. 4- Buscar 5- En Acciones: Eliminar
El usuario acepta el mensaje de confirmación.	NA	NA	NA	NA	NA	NA	El sistema muestra un mensaje de confirmación "Solicitud Eliminada Satisfactoriamente" y muestra en "Buscar" el listado actualizado de las Solicitudes	Satisfactorio	
El usuario cancela la acción.	NA	NA	NA	NA	NA	NA	El sistema cierra la ventana emergente y permanece en el mismo lugar	Satisfactorio	

Tabla 8 Matriz de Datos. Eliminar Solicitud de Calificación

SC 4 Modificar Solicitudes de Calificación

Escenario	V 1 Sol.	V 2 Fax	V 3 Teléf.	V 4 Correo	V 5 Nac.	V 6 F de S.	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
El usuario selecciona "Buscar", busca la Solicitud deseada y selecciona en acciones la opción modificar Solicitud	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	El Sistema muestra un panel con los datos a modificar.	*Satisfactorio*	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Calificacion de Solicitudes. 4- Buscar 5- Acciones: Modificar
El usuario modifica los datos en los campos correctamente.	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	El sistema modifica y actualiza la lista de esa solicitud.	*Satisfactorio*	
El usuario modifica los datos en los campos Incorrectamente.	*I*	*NA*	*NA*	*NA*	*I*	*I*	El sistema modifica y actualiza la lista de esa solicitud.	*No Satisfactorio*	

Tabla 9 Matriz de Datos. Modificar Solicitud de Calificación

SC 5 Mostrar Detalles de Solicitudes de Calificación

Escenario	V 1 Sol.	V 2 Fax	V 3 Teléf.	V 4 Correo	V 5 Nac.	V 6 F de S.	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
El usuario selecciona "Buscar", busca la Solicitud deseada y selecciona en acciones la opción Mostrar Detalles de las Solicitudes de Calificación.	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	El sistema muestra un panel con un formulario que contiene todos los datos de esa Solicitud.	*Satisfactorio*	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Calificacion de Solicitudes. 4- Buscar 5- Acciones: Mostrar Detalles de Solicitudes de Calificación.
El usuario Observa los Detalles de la Solicitud de Calificación y le da al botón de cerrar.	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	El sistema cierra el panel y permanece en el mismo lugar.	*Satisfactorio*	

Tabla 10 Matriz de Datos. Mostrar Detalles de Solicitud de Calificación

Los resultados de las pruebas que no fueron satisfactorias pasaron a ser no conformidades y se emitieron en el registro de defectos y dificultades detectados que se encuentra en la parte final de cada diseño de caso de prueba con los siguientes campos:

- Elemento: Se especifica el nombre del elemento.
- No: Se especifica el número de la no conformidad.
- No conformidad: Se describe la no conformidad.
- Aspecto correspondiente: Se especifica el aspecto correspondiente a la no conformidad.
- Etapa de detección: Se especifica la etapa de detección del error.
- Significativa: Se pone una (X), en caso de que la no conformidad estuviera clasificada como significativa.
- No significativa: Se pone una (X), en caso de que la no conformidad estuviera clasificada como no significativa.
- Recomendación: Se pone una (X), en caso de que la no conformidad solo fuera una recomendación.
- Estado NC: Se coloca el estado de la no conformidad y la fecha, cada vez que se revisa se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó [RA: Resuelta, PD: Pendiente, NP: No Procede].
- Respuesta del equipo de desarrollo: Esta columna se comienza a llenar a partir de la segunda iteración, y es responsabilidad del equipo de desarrollo, quien especifica si se ha resuelto o no la no conformidad y en caso de no proceder se explica la causa.

Registro de defectos y dificultades

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	EC 1.3 Los datos no válidos como números en el campo de Solicitante colapsan el sistema.	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Solicitudes de Calificación. 4- Adicionar	1ra etapa	X		X	10/02/10 PD: (Pendiente) 5/03/10 RA: Resuelta	
Aplicación	2	EC 4.3 El usuario modifica los datos en los campos Incorrectamente y el sistema modifica igual.	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Solicitudes de Calificación. 4- Buscar 5- Acciones: Modificar	1ra etapa	X		X	10/02/10 PD: (Pendiente) 6/03/10 RA: Resuelta	

Tabla 11 Registro de defectos y dificultades detectados

La plantilla de No Conformidades recoge además los errores que son detectados durante la revisión de la documentación del sistema. Se elabora un documento por cada revisión que se haga y se controlan a través de versiones según se vayan eliminando los errores, hasta que finalmente se hayan erradicado todos los defectos que posea el elemento que se prueba. Además de estar incluido en la planilla de diseño de casos de prueba, estas no conformidades se van registrando en un documento aparte para luego enviarlo al equipo de desarrolladores. De igual manera, aún trabajando paralelamente con los casos de prueba, el documento emitido a los desarrolladores es independiente, porque los casos de prueba son para consumo y único uso de los probadores.

Con los resultados de la ejecución de las pruebas de Caja Negra podemos decir que el caso de prueba Gestionar Solicitudes de Calificación en sentido general se evalúa satisfactoriamente, aunque se detectaron algunos fallos en la validación de campos de texto, por ende es necesario realizar un proceso de depuración de las faltas asociadas a los fallos identificados.

2.5 Diseño de Casos Pruebas de Caja Blanca. Técnica, Camino Básico

Esta técnica se basa en obtener una medida de la complejidad del diseño procedimental de un programa (o de la lógica del programa). Esta medida es la complejidad ciclomática de McCabe, y representa un límite superior para el número de casos de prueba que se deben realizar para asegurar que se ejecuta cada camino del programa.

Los pasos a realizar para aplicar esta técnica son:

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

Cada construcción lógica de un programa tiene una representación.

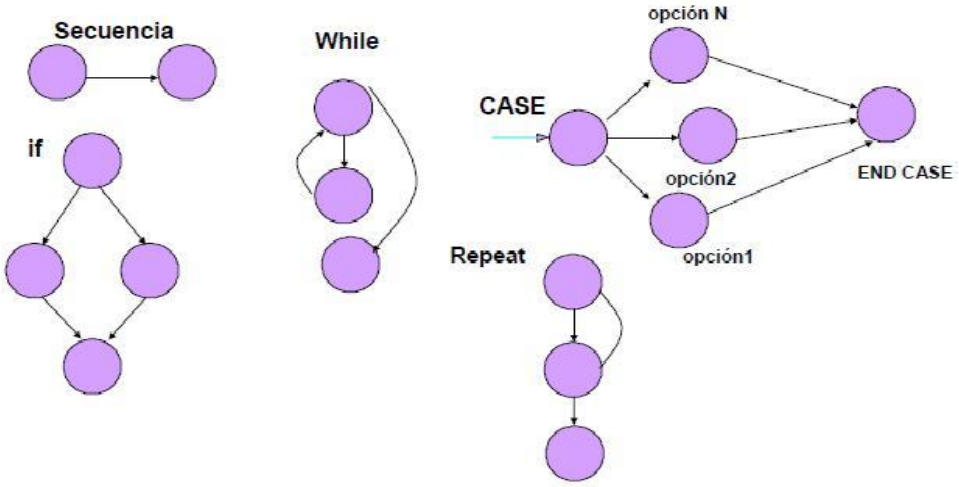

Figura 4 Representación en grafo de flujo de las estructuras lógicas de un programa

A continuación se detallan los pasos para aplicar la técnica de camino básico:

Representar el programa en un grafo de flujo

El grafo de flujo se utiliza para representar flujo de control lógico de un programa. Para ello se utilizan los tres elementos siguientes:

- *Nodos*: Representan cero, una o varias sentencias en secuencia. Cada nodo comprende como máximo una sentencia de decisión (bifurcación).
- *Aristas*: Líneas que unen dos nodos.
- *Regiones*: Áreas delimitadas por aristas y nodos. Cuando se contabilizan las regiones de un programa debe incluirse el área externa como una región más.
- *Nodos predicados*: Cuando en una condición aparecen uno o más operadores lógicos (AND, OR, XOR,...) se crea un nodo distinto por cada una de las condiciones simples. Cada nodo generado de esta forma se denomina nodo predicado.

La Figura 5 muestra un grafo de flujo del diagrama de módulos correspondiente. Nótese cómo la estructura principal corresponde a un *while*, y dentro del bucle se encuentran anidados dos sentencias *if*.

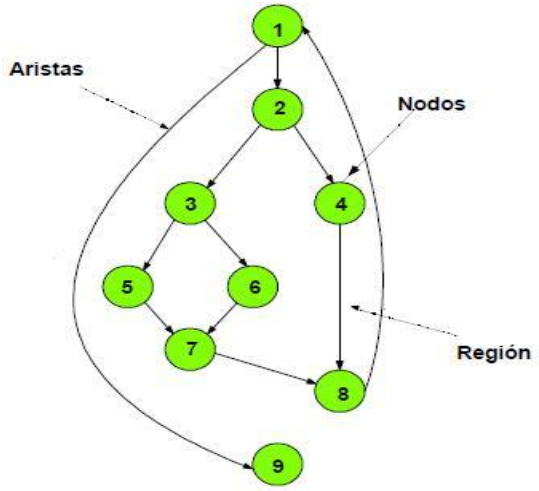

Figura 5 Ejemplo de grafo de flujo correspondiente a un diagrama de módulos

Calcular la complejidad ciclomática

Existen varias formas de calcular la complejidad ciclomática de un programa a partir de un grafo de flujo:

- El número de regiones del grafo coincide con la complejidad ciclomática, V (G).
- La complejidad ciclomática, V (G), de un grafo de flujo G se define como V (G) = Aristas – Nodos + 2.
- La complejidad ciclomática, V (G), de un grafo de flujo G se define como V (G) = Nodos Predicado + 1.

El resultado obtenido en el cálculo de la complejidad ciclomática define el **número de caminos independientes dentro de un fragmento de código** y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. (A Complexity Measure, 1976)

Una vez calculada la complejidad ciclomática de un fragmento de código, se puede determinar el riesgo que supone utilizando los rangos definidos en la siguiente tabla:

Complejidad Ciclomática	Evaluación del Riesgo
1-10	Programa Simple, sin mucho riesgo
11-20	Más complejo, riesgo moderado
21-50	Complejo, Programa de alto riesgo
50	Programa no testeable, Muy alto riesgo

Tabla 12 Riesgo del cálculo de la Complejidad Ciclomática

Esta complejidad ciclomática determina el número de casos de prueba que deben ejecutarse para garantizar que todas las sentencias de un programa se han ejecutado al menos una vez, y que cada condición se habrá ejecutado en sus vertientes verdadera y falsa. Veamos ahora, cómo se identifican estos caminos.

Determinar el conjunto básico de caminos independientes

El conjunto de caminos independientes de un grafo no es único. No obstante, a continuación, se muestran algunas heurísticas para identificar dichos caminos:

- Elegir un camino principal que represente una función válida que no sea un tratamiento de error. Debe intentar elegirse el camino que atraviese el máximo número de decisiones en el grafo.
- Identificar el segundo camino mediante la localización de la primera decisión en el camino de la línea básica alternando su resultado mientras se mantiene el máximo número de decisiones originales del camino inicial.
- Identificar un tercer camino, colocando la primera decisión en su valor original a la vez que se altera la segunda decisión del camino básico, mientras se intenta mantener el resto de decisiones originales.
- Continuar el proceso hasta haber conseguido tratar todas las decisiones, intentando mantener como en su origen el resto de ellas.

Este método permite obtener V (G) caminos independientes cubriendo el criterio de cobertura de decisión y sentencia.

Derivar los casos de prueba que fuerzan la ejecución de cada camino

El último paso es construir los casos de prueba que fuerzan la ejecución de cada camino.

Número del Camino	Caso de Prueba	Resultado Esperado

Tabla 13 Posible representación de casos de prueba para pruebas estructurales

2.5.1 Diseño de Prueba - CB del Caso de Uso: Gestionar Solicitud de Calificación

- **Adicionar Solicitudes de Calificación**

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

```
public function executeInsertar()
 {
```

```
1-  $nombre = $this->getRequestParameter("txtsolicitante");
 1-  $correo = $this->getRequestParameter("txtcorreo");
  1-  $fax = $this->getRequestParameter("txtfax");
  1-  $telf = $this->getRequestParameter("txttelefono");
  1-  $solicitante = TsolicitantepPeer::Insertar($nombre, $correo, $fax, $telf);
  1-  $fSol = $this->getRequestParameter('fSolicitud');
  1-   $solicitud = TsolicitudpPeer::Insertar($fSol, $solicitante->getIdsolicitante());
1-  $calificacion = TsolCalificacionPeer::Insertar($this->getRequestParameter("mnnacionalidad"));
  1-  $done = TsolicitudpPeer::GenerarId($solicitud, $calificacion, 'ca', $fSol);
  2-  if ($done)
  3-  return  $this->redirect ( 'calificacion/calificacion' );
        else
   4- return $this->msg = "Error insertando en la base de datos";
}
```

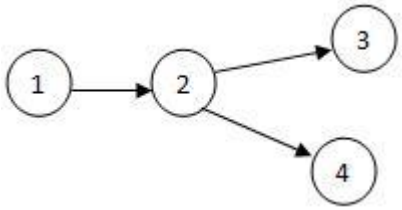

Figura 6 Grafo de flujo de datos para el método Adicionar Solicitudes de Calificación

Paso 2. Calcular la Complejidad Ciclomática

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

V (G) = 3 – 4 + 2

V (G) = 1

Paso 3. Determinar los Caminos Independientes

Caminos Básicos

CB1: 1 – 2– 3

CB2: 1 – 2– 4

Paso 4 Casos de Prueba del Camino Básico

CB1: 1 – 2– 3

CB2: 1 – 2– 4

Caso de Prueba: Gestionar Solicitudes de Calificación

Entrada: Solicitante, Fax, Teléfono, Correo Electrónico, Nacionalidad, Fecha de Solicitud.

Resultado Esperado: Se Adiciona la solicitud de Calificación.

Resultado de la prueba: Satisfactorio.

- **Modificar Solicitudes de Calificación**

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

```
public function executeModificar()
 {
1-        if ($this->getRequest ()->getMethod () == sfRequest::POST)
          {
 2-   $id = $this->getRequestParameter ('txtId');
       $solicitante = $this->getRequestParameter('txtSolicitante');
      $fax = $this->getRequestParameter('txtFax');
     $tele = $this->getRequestParameter('txtTelefono');
     $correo = $this->getRequestParameter('txtCorreo');
    $estado = $this->getRequestParameter ('txtEstado');
    $observaciones = $this->getRequestParameter ('txtObservaciones');
   $nacionalidad = $this->getRequestParameter ('txtNacionalidad');
   $fInscripcion = $this->getRequestParameter ('txtFechaIn');
   $fCertifico = $this->getRequestParameter ('txtFechaCe');
  $fDictamen=$this->getRequestParameter
('txtFechaDi');$solicitud=TsolCalificacionPeer::Modificar($id,$solicitante,$fax,$tele,$correo,$estado,
$observaciones,$nacionalidad,$fInscripcion,$fCertifico,$fDictamen);
  $idsol = $solicitud->getIdsolicitud();
2-  $libroca = TlibrocalificacionPeer::Todas();
3-  foreach ($libroca as $ca)
  {
 4-   if($ca->getIdsolicitud() == $idsol)
            {
```

```
5-        $this->redirect('calificacion/calificacion');
          break;
          }
 }
 6-  if ($solicitud->getEstado() == "Aprobado")

7-     $this->redirect('lregistro/registrarca?id='.$idsol);
else
8-     $this->redirect('calificacion/calificacion');
 }
9-   else return sfView::SUCCESS;
 }
```

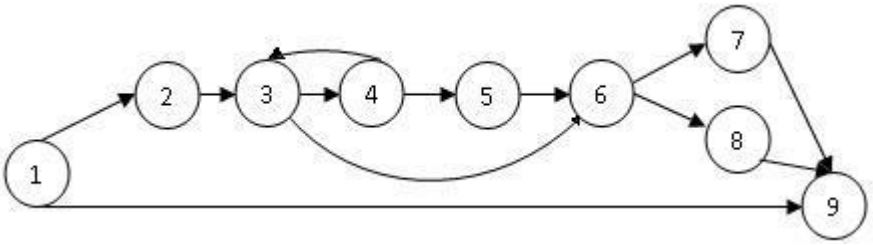

Figura 7 Grafo de flujo de datos para el método Modificar Solicitudes de Calificación

Paso 2. Calcular la Complejidad Ciclomática

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

V (G) = 12 – 9 + 2

V (G) = 5

Paso 3. Determinar los Caminos Independientes

Caminos Básicos

CB1: 1 – 2– 3 – 4 – 5 – 6 – 7 – 9

CB2: 1 – 2– 3 – 4 – 5 – 6 – 8 – 9

CB3: 1 – 2– 3 – 6 – 7 – 9

CB4: 1 – 2– 3 – 6 – 8 – 9

CB5: 1 – 9

Paso 4 Casos de Prueba del Camino Básico

CB1: 1 – 2– 3 – 4 – 5 – 6 – 7 – 9

CB2: 1 – 2– 3 – 4 – 5 – 6 – 8 – 9

Caso de Prueba: Gestionar Solicitudes de Calificación

Entrada: Solicitante, Fax, Teléfono, Correo Electrónico, Estado, Nacionalidad, Fecha de Certifico, Fecha Dictamen, Fecha de Solicitud.

Resultado Esperado: Se Modifica la solicitud de Calificación.

Resultado de la prueba: Satisfactorio.

- **Buscar Solicitudes de Calificación**

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

```
public function executeMostrarcalificacion()
  {
    1- $nacionalidad = TsolCalificacionPeer::Nacionalidades();
       $texto = $this->getRequestParameter('txtTexto');
       $offset = $this->getRequestParameter('txtInicio');
       $limite = $this->getRequestParameter('txtCantidad');
       $a = TsolCalificacionPeer::BuscarTexto($texto, $limite, $offset);
       $t = TsolCalificacionPeer::CantidadTexto($texto);
       $array_json = '{}';
       $Json = array();
    1- $i = 0;
    2- foreach ($a as $elem)
       {
    3-   foreach ($nacionalidad as $nac)
                {
    4-            if ($nac->getIdelemento() == $elem->getIdnacionalidad())
                  {
    5-            $nomnac = $nac->getNombre();
                  break;
                  }
```

```
                }
6-              $id = $elem->getIdsolicitud();
6-              $Json[$i++] = array (  //"id" => $elem->getIdsolicitud(),
                "id" => "$id",
                "Identificador" => $elem->getIdsolicitud(),
                "Solicitante" => $elem->getTsolicitudp()->getTsolicitantep()->getNombre(),
          "Fecha de Solicitud" => $elem->getTsolicitudp()->getFechasolicitud(),
                "Estado" => $elem->getTsolicitudp()->getEstado(),
                "Nacionalidad" => $nomnac  ) ;
          }
7-  $titulos = array( 0 => "Identificador" ,  1 => "Solicitante",  2 => "Fecha de Solicitud",
                3 => "Estado",  4 => "Nacionalidad",  5 => "Acciones", 6 => "1",  7 => "2",
                8 => "3");
    $acciones = array( array("funcion" => "ajaxModificarjs",
                      "img" => '../../../../images/icon/edit.png" title="Modificar Solicitud"'),
                    array("img" => '<img src="../../../../images/icon/delete.png" title="Eliminar
Solicitud"',  "confirmar" => "1", "accion" => "/concesionariop.php/calificacion/eliminar",
                      "mensaje" => "Est&aacute; seguro que desea eliminar la solicitud",
                           "titulo" => "Eliminar Solicitud" ), array("funcion" => "ajaxDetallesjs",
       "img" => '../../../../images/icon/filter.png" title="Detalles Solicitud"') );
    $Json[$i++]= $acciones;
                $Json[$i++] = $titulos;
               $Json[$i++] = array("total" => $t);
 7-          $array_json = json_encode($Json);
 8-         $this->getResponse()->setHttpHeader("application/json");
               echo $array_json;
  8 -        return sfView::NONE;
  }
```

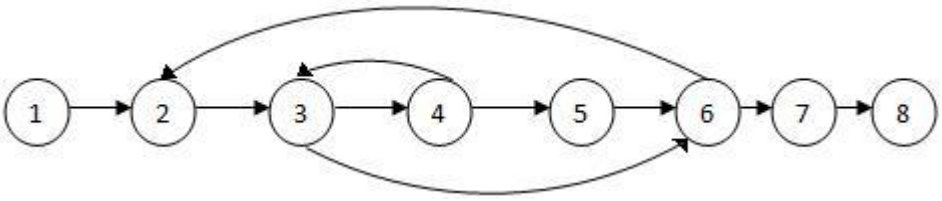

Figura 8 Grafo de flujo de datos para el método Buscar Solicitudes de Calificación

Paso 2. Calcular la Complejidad Ciclomática

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

V (G) = 10 – 9 + 2

V (G) = 3

Paso 3. Determinar los Caminos Independientes

Caminos Básicos

CB1: 1 – 2– 3 – 4 – 5 – 6 – 7 – 8

CB2: 1 – 2– 3 – 6 –– 7 – 8

Paso 4 Casos de Prueba del Camino Básico

CB1: 1 – 2– 3 – 4 – 5 – 6 – 7– 8

CB2: 1 – 2– 3 – 6 –– 7– 8

Caso de Prueba: Gestionar Solicitudes de Calificación

Entrada: Identificador, Solicitante, Fecha de Solicitud, Estado, Nacionalidad.

Resultado Esperado: Se Busca la solicitud de Calificación.

Resultado de la prueba: Satisfactorio.

- **Eliminar Solicitudes de Calificación**

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

```
public function executeEliminar()
  {
    1-   $id = $this->getRequestParameter ('id', null);
              $array_json = '{}';
      1- $Json = array();
        2-try
```

```
        {
    3 -TsolicitudpPeer::Eliminar($id); $Json[0] = array('estado'=> true, 'men'=>'Solicitud
Eliminada Satisfactoriamente' );
        }
    4-      catch (Exception $e)
        {
    5-      $Json[0] = array('estado'=> false, 'men'=> $e->getMessage());
        };
        6- $array_json = json_encode($Json);
        7-      $this->getResponse()->setHttpHeader("application/json");
          echo $array_json;
        7-      return sfView::NONE;
}
```

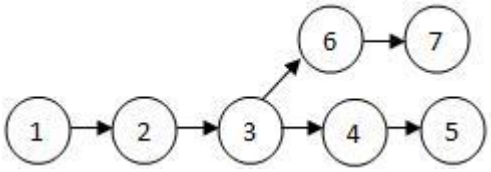

Figura 9 Grafo de flujo de datos para el método Eliminar Solicitudes de Calificación

Paso 2. Calcular la Complejidad Ciclomática

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

V (G) = 6 – 7 + 2

V (G) = 1

Paso 3. Determinar los Caminos Independientes

Caminos Básicos

CB1: 1 – 2– 3 – 6 -7

CB2: 1 – 2– 3 – 4– 5

Paso 4 Casos de Prueba del Camino Básico

CB1: 1 – 2– 3 – 6 -7

CB2: 1 – 2– 3 – 4– 5

Caso de Prueba: Gestionar Solicitudes de Calificación

Entrada: Ninguna

Resultado Esperado: Se Elimina la solicitud de Calificación.

Resultado de la prueba: Satisfactorio.

- **Mostrar Detalles de Solicitudes de Calificación**

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

```
public function executeAjaxDetalles() // Mostrar Detalles de una Solicitud
 {
1-   if($this->getRequest()->getMethod() != sfRequest::POST)
        {
2-              $this->redirect('calificacion/calificacion#buscar');
        }
3-      else
        {
4-              $array_json = '{}';
                $id = $this->getRequestParameter('solicitud');
            $ca = TsolCalificacionPeer::getbyId($id);
                $solicitudJson = array('solicitante'=> $ca->getTsolicitudp()->getTsolicitantep()->getNombre(), 'fechas'=> $ca->getTsolicitudp()->getFechasolicitud(),'estado'=> $ca->getTsolicitudp()->getEstado(),'nacionalidad' => TsolCalificacionPeer::NacionalidadSol($ca),
'categoria' => TsolCalificacionPeer::CategoriaCa($ca),'tomo'=> $ca->getTsolicitudp()->getTomo(),'folio'=> $ca->getTsolicitudp()->getFolio(),'asiento'=> $ca->getTsolicitudp()->getAsiento(), 'fechain'=> $ca->getTsolicitudp()->getFechainscripcion(),
 'fechace'=> $ca->getTsolicitudp()->getFechacertifico(),'fechadi'=> $ca->getFechaDicatamen(),
 'observ'=> $ca->getTsolicitudp()->getObservaciones(), );

                $array_json = json_encode($solicitudJson);

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

                echo $array_json;

5-              return sfView::NONE;

        }
 }
```

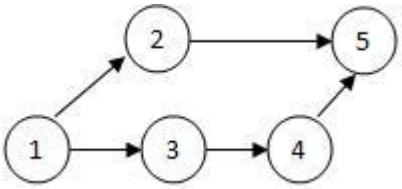

Figura 10 Grafo de flujo de datos para el método Mostrar Detalles de Solicitudes de Calificación

Paso 2. Calcular la Complejidad Ciclomática

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

V (G) = 5–5 + 2

V (G) = 2

Paso 3. Determinar los Caminos Independientes

Caminos Básicos

CB1: 1 – 2 - 5

CB2: 1 – 3 – 4 - 5

Paso 4 Casos de Prueba del Camino Básico

CB1: 1 – 2 - 5
CB2: 1 – 3 – 4 - 5

Caso de Prueba: Gestionar Solicitudes de Calificación

Entrada: Ninguna

Resultado Esperado: Se Muestran los Detalles de la solicitud Calificación.

Resultado de la prueba: Satisfactorio.

Conclusiones parciales

En este capítulo se realizaron pruebas a la documentación y aplicación, utilizando listas de chequeo, técnicas y métodos de pruebas, mediante los casos de pruebas diseñados anteriormente, documentando los errores encontrados en el documento de No Conformidades. Se realizaron los artefactos que estaban definidos como entregables, dentro de ellos el Plan de Pruebas, Diseños de Casos de Pruebas tanto de caja negra como de caja blanca. Se realizó una revisión completa al sistema tanto a la interfaz como al código, con la intención de una mejor organización de la ejecución de las pruebas.

Capítulo 3: Análisis de los Resultados.

Introducción

Cada proceso de pruebas demanda de un análisis final de los errores obtenidos, de forma tal que se proporcione una evaluación del producto que se prueba teniendo en cuenta el seguimiento que se ha ido registrando de los defectos y fallos del sistema durante todo el proceso. En este capítulo se procura analizar las no conformidades encontradas según el elemento probado y técnica empleada. Se hace una valoración del producto de acuerdo con los atributos de calidad como la seguridad, usabilidad, portabilidad, fiabilidad y confiabilidad, basándose en los resultados obtenidos al aplicar listas de chequeo y los diseños de caso de pruebas desarrollados para la funcionalidad del módulo Registro Petrolero.

3.1 Resultados obtenidos al aplicar las pruebas al módulo Registro Petrolero

Los resultados obtenidos en el módulo Registro Petrolero de manera general se clasifican como satisfactorios, pues se encontraron algunas fallas o defectos en la aplicación del diseño de casos de pruebas realizados y pruebas a la documentación de dicho módulo.

3.2 Análisis de las No Conformidades de la Documentación

Una vez registradas todas las no conformidades encontradas (ver el documento de No Conformidades) durante la revisión de la documentación, tomando entre sus documentos más significativos el de Especificación de Requisitos. Siendo este uno de los aspectos de mayor importancia a la hora de entrega a los clientes. Se tomaron en cuenta una serie de aspectos para la revisión del mismo entre los que se encuentran:

- Ortografía
- Redacción
- Correspondencia con otra documentación
- Formato
- Error técnico

Como resultado de lo antes expuesto se puede afirmar que se detectaron dificultades en dicho documento de Especificación de Requisitos, señalando que la ayuda que debería tener la aplicación no funciona 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. Los errores en la documentación pueden llegar a ser tan perjudiciales para la aprobación del producto, como los errores en los datos o en el código fuente, por eso es de vital importancia la prueba a la documentación, ya que es un elemento clave para la aceptación del software por parte de los usuarios.

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

3.2.1 Especificación de Requisitos

	No	No conformidad	Aspecto correspondiente	Etapa	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Documentación Especificación de Requisitos	1	Los requisitos para la documentación de usuarios en línea y ayuda aún no funcionan.	Documento Especificación de Requisitos, epígrafe 8 página 16.	Pruebas a la Documentación.	X		El Manual de Usuario y la ayuda en la aplicación es necesario que se realice y funcione.	20/02/10 PD: (Pendiente)	
Documentación Especificación de Requisitos	2	Especificar que ya no es **polo** sino **centro**. (GEySED)	Documento Especificación de Requisitos, pagina principal.	Pruebas a la Documentación.		X	Revisar y cambiar **polo** por **centro** de Geoinformática y Señales Digitales.	20/02/10 PD: (Pendiente)	

Tabla 14 No Conformidades en el documento de Especificación de Requisitos

3.3 Análisis de las No Conformidades de la Aplicación

Para clasificar las no conformidades en significativas de la aplicación se tuvieron en cuenta los siguientes criterios:

- Ortografía
- Funcionalidad
- Validación
- Excepciones

- Correspondencia con documentación
- Opciones que no funcionan

La realización de las pruebas es indiscutiblemente la mejor forma de conocer el grado de complejidad a que ocurra un error antes de ponerlo en manos de los clientes. En este caso las pruebas fueron de gran utilidad debido a que arrojaron como resultado algunos errores que no se habían detectado por los desarrolladores y que era necesario erradicar.

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

3.3.1 Caso de Prueba Gestionar Solicitudes de Calificación

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	EC 1.3 Los datos no válidos como números en el campo de Solicitante colapsan el sistema.	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Solicitudes de Calificación. 4- Adicionar	1ra etapa	X		X	10/02/10 PD: (Pendiente) 5/03/10 RA: Resuelta	
Aplicación	2	EC 4.3 El usuario modifica los datos en los campos Incorrectamente y el sistema modifica igual.	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Solicitudes de Calificación. 4- Buscar 5- Acciones: Modificar	1ra etapa	X		X	10/02/10 PD: (Pendiente) 6/03/10 RA: Resuelta	

Tabla 15 No Conformidades en el CU - Gestionar Solicitud de Calificación

3.3.2 Caso de Prueba Gestionar Solicitudes de Permiso de Explotación

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativ	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	EC 1.3 El usuario introduce datos en los campos de Perforación en **Mar** correctamente.	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Solicitudes de Permiso de Explotación. 4- Adicionar. Opción: **Mar**	1ra etapa	X		Validar el botón Adicionar. No Adiciona en esta opción.	10/02/10 PD: (Pendiente) 13/03/10 RA: Resuelto	
Aplicación	2	EC 4.2 El usuario modifica los datos en los campos correctamente.	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Solicitudes de Permiso de Explotación. 4- Buscar 5- Acciones: **Modificar**	1ra etapa	X		El sistema debería de dar la opción de entrar el **Municipio** que no lo pone en el desplegable.	10/02/10 PD: (Pendiente) 5/03/10 RA: Resuelto	
Aplicación	3	EC 4.3 El usuario modifica los datos en los campos Incorrectamente.	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Solicitudes de Permiso de Explotación. 4- Buscar 5- Acciones: **Modificar**	1ra etapa	X		El sistema no debería modificar los datos si estos son incorrectos. Validar los campos de textos.	PD: (Pendiente)	

Tabla 16 No Conformidades en el CU - Gestionar Solicitudes de Permiso de Explotación

3.3.3 Caso de Prueba Gestionar Documentos

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	EC 5.2 El usuario puede ver el documento pero colapsan el sistema.	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Gestión. 3-Documentos. 4- Buscar 5- Acciones: Mostrar Documento.	1ra etapa	X		Ver por que explota el sistema cuando el usuario ve algún documento y no puede seguir trabajando en la aplicación	10/02/10 PD: (Pendiente) 5/03/10 RA: Resuelto	

Tabla 17 No Conformidades en el CU - Gestionar Documentos

3.3.4 Caso de Prueba Generar Documentos

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	EC 1.3 El usuario pincha en el botón de Generar el documento y no hace nada.	- Página Principal - 1- El usuario selecciona el módulo Registro Petrolero. 2- Selecciona Consultas. 3-Generar Documentos. 4- Documento por solicitudes.	1ra etapa	X		X	11/02/10 PD: (Pendiente) 5/03/10 RA: Resuelta	

Tabla 18 No Conformidades en el CU - Generar Documentos

3.4 Principales errores Detectados

En esta primera etapa de aplicación de las pruebas se encontraron 9 no conformidades al módulo Registro Petrolero, dentro de los principales errores encontrados se tienen los siguientes:

- Entrada de datos no válidos al sistema.
- Validar los botones. Ejemplo, el botón de adicionar solicitudes de explotación de Mar y Modificar.
- Rectificar que ya no es polo sino centro.
- La ayuda en la aplicación no funciona.
- Existe una funcionalidad que se llama Cantidad de Solicitudes que no se encuentra en la documentación de especificación de requisitos.
- Hay opciones que no funcionan como: La lista desplegable Municipio (combobox), del caso de uso Gestionar Solicitudes de Permiso de Perforación por Mar.
- En la aplicación se detectaron algunas faltas ortográficas. Ejemplo de esto es el nombre del campo Correo Electrónico que está sin tilde en el caso de uso Gestionar Solicitudes de Calificación, el (radiobutton) de Perforación que lleva tilde en el caso de uso Gestionar Solicitudes de Permiso de perforación, en insertar prórroga el campo de texto de prórroga no lleva la tilde.

3.5 Resultados al aplicar Listas de Chequeo

La lista de chequeo usada para la evaluación, cuenta con un conjunto de preguntas relacionada a cada atributo de calidad, a estas se le asignaron un peso según su importancia y un valor según el cumplimiento de la pregunta en el sistema.

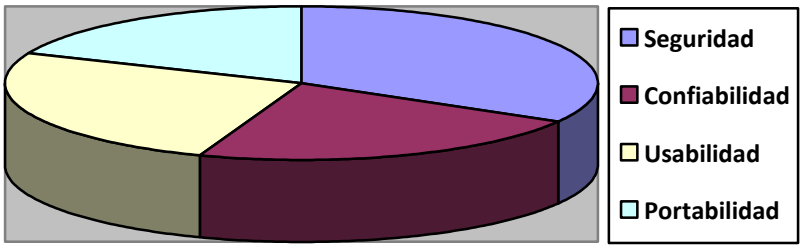

Figura 11 Atributos de Calidad

A partir de los resultados cuantitativos obtenidos puede calificarse el producto como:

- Muy Seguro

Por ser un sistema que accedan solamente a la aplicación las personas previamente autorizadas y éstas puedan realizar únicamente las funcionalidades definidas con su rol. Toda la información que manipula está protegida contra el uso no autorizado y tiene implementada técnicas para la encriptación de la información que se intercambia entre el servidor y los clientes. La seguridad indica que un sistema está libre de peligro, daño o riesgo. Se entiende como peligro o daño todo aquello que pueda afectar su funcionamiento directo o los resultados que se obtienen del mismo. Un sistema es seguro cuando cumple con estas características:

- Integridad: La información solo puede ser modificada por quien está autorizado y de manera controlada.
- Confidencialidad: La información es accedida solamente por personas autorizadas.
- Disponibilidad: Debe estar disponible cuando se necesita.

- Muy Confiable

Por ser un sistema que responde tal y como lo espera el usuario, es una propiedad que es igual a su fidelidad. Esta significa el grado de confianza del usuario en el sistema, que no fallará al utilizarlo normalmente. Brinda disponibilidad, lo que garantiza al usuario poder contar con los servicios de este en

cualquier momento que lo necesite. La confiabilidad está estrechamente relacionada con la: Disponibilidad, Fiabilidad, Seguridad y Protección. El funcionamiento de un sistema seguro depende normalmente de que el sistema esté disponible y su funcionamiento sea fiable.

- Usable

El sistema debe poder ser usado por cualquier persona que posea conocimientos básicos de computación. La Aplicación cuenta con una ayuda para facilitar su uso además de la documentación. Posee una interfaz amigable, que contiene los colores característicos de la ONRM. La información deberá estar disponible en todo momento, limitada solamente por las restricciones que esta tenga de acuerdo con las políticas de seguridad del sistema.

- Portabilidad

Está característica que posee el software indica que es multiplataforma, el código fuente se puede utilizar tanto para Linux como para Windows, no es necesario crear un nuevo código cuando el sistema se ejecute en una plataforma u otra. Este es una de las características claves de la programación de alto nivel. A mayor portabilidad menor es la dependencia del software con respecto a la plataforma.

Conclusiones Parciales

Al terminar este capítulo se llegó a la conclusión de que el proceso de pruebas que fue llevado a cabo en el módulo Registro Petrolero se realizó satisfactoriamente, ya que a estas se le detectaron algunos errores que fueron documentados para su previa eliminación, y seguidamente pase a la fase de pruebas de aceptación por el usuario. Se probaron todos los casos de uso, verificando que todos los requisitos planteado por el cliente funciona.

Conclusiones.

Toda la labor realizada en el presente trabajo de diploma al planificar y aplicar las pruebas de software al módulo Registro Petrolero se realizó exitosamente. Cumpliendo todos los objetivos planteados para esta investigación se obtuvieron los siguientes resultados:

- Se verificó la funcionalidad de los requisitos planteados por los clientes de la Oficina Nacional de Recursos Minerales, mediante la aplicación de la estrategia seleccionada. Basada en técnicas de pruebas de caja negra: partición equivalente, de caja blanca: camino básico, y además, las pruebas a la documentación, aplicada al módulo Registro Petrolero, perteneciente al Sistema de gestión de Datos Geológicos.

- Se detectaron algunos errores que no fueron visto por los programadores del producto mediante la aplicación de casos de pruebas de caja negra al módulo Registro Petrolero.

- Se logró una mayor organización y planificación del proceso de pruebas efectuado, al módulo Registro Petrolero, lo cual se obtuvo mediante la elaboración del Plan de Pruebas.

- Se ejecutaron y evaluaron 9 casos de pruebas diseñados y se obtuvo la lista de defectos con un resumen de errores y sugerencias.

De forma general, el flujo de trabajo de pruebas se efectuó exitosamente partiendo de que el éxito de las pruebas se encuentra en la detección de errores. Todos los errores encontrados fueron informados y documentados para su eliminación, para que así quede libre de defectos el módulo Registro Petrolero que forma parte del SGDG dentro de la instancia productiva del centro GEySED.

Recomendaciones.

Posterior a la realización de todas pruebas concebidas para el módulo Registro Petrolero, y dados los resultados obtenidos durante este proceso, se recomienda que:

- Aplicar las pruebas al sistema en próximas iteraciones.
- Aplicar las pruebas de Carga y Estrés para un mejor funcionamiento y aseguramiento del sistema.

Referencias Bibliográficas

ONRM. 2009. Oficina Nacional de Recursos Minerales. [En línea] 2009. [Citado el: 2009.] http://www.onrm.minbas.cu/.

Vinent, Manuel Iturralde. 2005. Primera Convención Cubana de Ciencias de la Tierra. Ciencia y Tecnología. 25 de 3 de 2005.

Sommerville, Ian. 2005. Ingeniería del software. Séptima edición. Madrid (España): Pearson Educación. S.A., 2005.

Carrasco Oscar M. Fernández. Un enfoque actual sobre la calidad del software [Publicación periódica]. - 1995.

Pressman, Roger S. 2009. Ingeniería de Software Un Enfoque Práctico. 2009.

Ambler, Scott W. 2004. [En línea] 2004. http://www.ambysoft.com/essays/flootSpanish.html.

JACOBSON, Ivar, RUMBAUGH, James y BOOCH, Grady. 2002. El lenguaje unificado de modelado. 2002.

BRUEGGE, Bernd, DUTOIT, Allen. 2002. Ingeniería de Software Orientado a Objetos. S.l.: Prentice Hall-Pearson Educación, 2002. págs. Cap. 9 páginas 326-369.

PRESSMAN, Roger. 2002. Ingeniería del Software. Un enfoque práctico. 2002.

Disponible en: http://bibliodoc.uci.cu/pdf/reg02689.pdf

PRESSMAN, Roger. 2004. Ingeniería del Software. Un enfoque práctico. Quinta edición. 2004.

IEEE. Std. 610-1990. Calidad de Software. [Online] Std. 610-1990. http://dmi.uib.es/~bbuades/calidad/calidad.PPT.

Ronquillo, Ing. Dayris Espinosa. 2009. Arquitectura de Software. 2009.

Daily. 2008. Definición. De. [En línea] Wordpress, 2008. http://definicion.de/modelo-de-calidad/.

*A Complexity Measure***. McCABE, Artículo original de THOMAS J. 1976.** NO.4, 1976, Vols. SE-2. URL: http://www.literateprogramming.com/mccabe.pdf

Fernández, Vladimir Martell. 2009. *PROYECTO TÉCNICO*. 2009.

MINBAS. 2007. *"MINISTERIO DE LA INDUSTRIA BASICA*.". [En línea] 2007. Disponible en: http://www.cubagob.cu/des_eco/minbas.htm.

ALARCÓN, Armando Silva. 2004. Modelos de calidad. La industria del software en México. Entérate. [En línea] Universidad Nacional Autónoma de México, 01 de 2004. [Citado el: 04 de 11 de 09.] Disponible en: http://www.enterate.unam.mx/Articulos/2004/Enero/modelos.htm.

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

Hower, Rick. 2010. SoftwareQATest.com. Web Site Test Tools and Site Management Tools. [En línea] 2010. http://www.softwareqatest.com/qatweb1.html.

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

Lovelle, Juan Manuel Cueva. 1999. Calidad del Software. [Online] 10 21, 1999. http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Calidad_software.PDF.

Bibliografía

Betancur, José Alejandro. 2009. slideshare. [En línea] 2009. http://www.slideshare.net/chiky/el-rol-de-pruebas-presentation.

ONRM. 2009. Oficina Nacional de Recursos Minerales. [En línea] 2009. [Citado el: 2009.] http://www.onrm.minbas.cu/.

Vinent, Manuel Iturralde. 2005. Primera Convención Cubana de Ciencias de la Tierra. Ciencia y Tecnología. 25 de 3 de 2005.

Sommerville, Ian. 2005. Ingeniería del software. Séptima edición. Madrid (España): Pearson Educación. S.A., 2005.

Carrasco Oscar M. Fernández. Un enfoque actual sobre la calidad del software [Publicación periódica]. - 1995.

Pressman, Roger S. 2009. Ingeniería de Software Un Enfoque Práctico. 2009.

Ambler, Scott W. 2004. [En línea] 2004. http://www.ambysoft.com/essays/flootSpanish.html.

Ronquillo, Ing. Dayris Espinosa. 2009. Arquitectura de Software. 2009.

*A Complexity Measure***. McCABE, Artículo original de THOMAS J. 1976.** NO.4, 1976, Vols. SE-2. URL: http://www.literateprogramming.com/mccabe.pdf

Fernández, Vladimir Martell. 2009. *PROYECTO TÉCNICO*. 2009.

Hower, Rick. 2010. SoftwareQATest.com. Web Site Test Tools and Site Management Tools . [En línea] 2010. http://www.softwareqatest.com/qatweb1.html.

Daily. 2008. Definición. De. [En línea] Wordpress, 2008. http://definicion.de/modelo-de-calidad/.

Pressman, Roger S. 2009. Ingeniería de Software Un Enfoque Práctico. 2009.

BRUEGGE, Bernd, DUTOIT, Allen. 2002. Ingeniería de Software Orientado a Objetos. S.l.: Prentice Hall-Pearson Educación, 2002. págs. Cap. 9 páginas 326-369.

JACOBSON, Ivar, RUMBAUGH, James y BOOCH, Grady. 2002. El lenguaje unificado de modelado. . 2002.

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

PRESSMAN, Roger. 2002. Ingeniería del Software. Un enfoque práctico. 2002.

Glosario de Términos

ONRM: Oficina Nacional de Recursos Minerales.

MINBAS: Ministerio de la Industria Básica.

PNICG: Programa Nacional de Informatización del Conocimiento Geológico.

UCI: Universidad de las Ciencias Informáticas.

SGDG: Sistema de Gestión de Datos Geológicos.

GEySED: Centro Geoinformática y Señales Digitales.

RUP: Proceso Unificado de Desarrollo.

UML: Lenguaje Unificado de Modelado.

CASE: Computer Aided Software Engineering. Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas informáticos, completamente o en alguna de sus fases.

AVL: Análisis de valores de límites.

CN: Caja Negra.

CB: Caja Blanca.

CU: Caso de Uso: Secuencias de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de las secuencias. Según RUP.

V (G): Complejidad ciclomática de McCabe: Se basa en el diagrama de flujo determinado por las estructuras de control de un determinado código. De dicho análisis se puede obtener una medida cuantitativa de la dificultad de crear pruebas automáticas del código y también es una medición orientativa de la fiabilidad del mismo. Es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Es una de las métricas de software de mayor aceptación, ya que ha sido concebida para ser independiente del lenguaje.