Diseño y Aplicación de casos de prueba al Sistema para Manejo Integral de Perforación de Pozos (SIPP).

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA.

AUTOR: Yisell Cruz Arias.

TUTOR: Ing. Ismaila López Sotolongo.

COTUTOR: Ing. Alexey Díaz Domínguez.

Ciudad de La Habana, Junio del 2010.
"Año del 52 de la Revolución"

Declaración de autoría

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

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

Autora: Yisell Cruz Arias.

_____

Tutor: Ismaila López Sotolongo.

_____

Agradecimientos

Quisiera agradecer a todas las personas que han estado a mi lado, ayudándome y brindándome amor, apoyo y amistad.

Ante todo agradezco a mi mamá, por serlo todo para mí, por su dedicación, amor, paciencia y tantas cualidades que me hacen esforzarme día a día para que se sienta orgullosa de mí.

Agradezco a mi abuelita Gloria; que aunque no esté junto a mí, el pensar en ella me hace ser mejor persona.

Quisiera agradecer a mi familia, mis tíos Alicia, Irma e Idael que han estado conmigo en estos 5 años de universidad y me han brindado mucho apoyo.

A mis amigos Iván, Yisel, Liana, Yorgenis, Mayrolis, por su amistad, paciencia y comprensión.

A mi novio por el amor y la comprensión en estos últimos años y llenar mi corazón en los momentos más difíciles de mi vida.

A mi hermana darme consejos y ayudarme.

A mis compañeros del grupo 9105 por un gran comienzo.

A mis compañeros del grupo 9306 por compartir alegrías, tristezas.

A mis viejos y nuevos amigos

A todos los profesores que tuve en estos cinco años, gracias por brindarme cada día, con paciencia y esmero sus conocimientos.

A todos aquellos que de alguna manera han transitado por mi vida y han dejado en ella una huella indestructible que no se borrará con el paso del tiempo.

A todos, muchísimas gracias.

Dedicatoria

A mi mamá, mi abuela, mi amor. ❤

Isela, Gloria, Alexey.

A mis sobrinos.

Miguel Enrique y Miguel Angel.

Resumen

En las últimas décadas se ha experimentado un crecimiento vertiginoso a nivel mundial en la industria del software. En la actualidad, la mayoría de las empresas demandan automatizar procesos de vital importancia que son claves para su correcto funcionamiento, lo que ha desatado un gran auge de las empresas que producen y exportan software exitosamente a lo largo del mundo, unido a una competencia cada vez más reñida en el mercado de productos de software.

La UCI, constituye en la industria cubana del software uno de los pilares de mayor importancia para la economía del país. Las ideas de convertir nuestros servicios informáticos en un rubro exportable, se ven frenadas por diversos problemas que atentan contra la calidad del proceso productivo y por consiguiente contra los resultados del producto final. Entre los productos desarrollados por la UCI se encuentra el Sistema de Manejo Integral de Perforación de Pozos, el producto está destinado a la Industria del Petróleo Cubano, específicamente a los pozos de petróleo en perforación, con el objetivo de gestionar todo el flujo de la información que es generado en estos. El propósito fundamental es que una vez puesto en ejecución este trabajo, se eleve la eficiencia y calidad en el control del flujo de información de estas empresas.

El presente trabajo surge a partir de la necesidad de controlar la calidad del producto SIPP, para detectar y eliminar el mayor número de fallas existentes en el mismo y obtener una aplicación con un mínimo de errores. Para dar cumplimiento a la idea planteada anteriormente, se diseñó una estrategia de prueba, teniendo como objetivos principales la planificación de las pruebas mediante la realización del plan de pruebas, el diseño de casos de pruebas, registrar los resultados en la lista de no conformidades, el análisis de los resultados obtenidos y concluir con una valoración del sistema.

Abstract

The worldwide advance of computerization goes together with the exponential demand of software, this way product quality becomes nearly a key point for developers taking into account its importance. Cuba recently has begun to take part of this market due to the economical outlook it offers. Software testing is the fundamental activity in the quality control performed to software development projects. Since the creation of its quality groups, the University of Computer Science (UCI) has been taking steps to progress in that field.

Within the set of projects hosted by UCI, Drilling Integrated Management System is being developed, a product intended to the Cuban Oil Industry, specifically for drilling oil wells, in order to allow the management of information generated in it. Our main purpose once the project is been implemented, is rising information flow control quality and efficiency in these companies.

This work focuses on the design and software testing implementation of SIPP product, so as to eliminate the most of existing bugs and this way obtain an application with a minimum of errors and thus with higher quality. To achieve this objective a test strategy was laid out to guide the entire testing design and implementation, analyzing results from each test, concluding with the system assessment

Índice

Índice de Figuras

Índice de Tablas

Introducción

El vertiginoso desarrollo actual de las tecnologías de la información y las comunicaciones, ha acrecentado la necesidad de las compañías de destinar una mayor cantidad de sus presupuestos a las nuevas tecnologías aplicadas al desarrollo de su actividad. Todas estas innovaciones tecnológicas ameritan una exigente evaluación de la calidad de los productos. La calidad constituye uno de los problemas que actualmente enfrenta la industria del software mundial, dicho aspecto incentiva la preocupación de muchos productores de software para obtener un producto fiable, eficaz, seguro y funcionalmente operativo que cumpla con los requerimientos del cliente.

Cuba se ha visto inmersa en esta carrera de producción de software teniendo como uno de sus principales exponentes a la Universidad de Ciencias Informáticas (UCI), la cual surge al calor de la batalla de ideas y tiene como misión fundamental crear software con una mejor calidad para la industria cubana y demás países que soliciten nuestros servicios. Para el cumplimiento de este objetivo se crean estructuras productivas que desarrollan líneas de investigación y la forma de automatizarlas, llevando consigo un gran nivel de responsabilidad para el logro de un producto con un óptimo nivel de calidad, por tanto se hace necesario implementar estrategias y correctos procesos de pruebas, que permitan satisfacer al cliente con un buen producto y a la vez ganar en tiempo y esfuerzo.

Uno de los productos con que cuenta la universidad es: Sistema de Manejo Integral de Perforación de Pozos (SIPP), que está destinado a los pozos de petróleo en perforación, con el propósito de gestionar todo el flujo de la información que es generado en estos. Al realizar un análisis del proceso de desarrollo de software se detectaron desavenencias que pueden inducir fallas en el producto final, como la poca especialización del negocio de perforación petrolera, que conlleva a errores en el negocio y el manejo de la información necesaria para la realización del producto, provocando el aumento de tiempo y costo, tanto en la corrección de errores, como en el tiempo de desarrollo. Parte de lo anteriormente planteado sucede por la ausencia de mecanismos para la gestión de la calidad en el proyecto que realiza este sistema. Por los problemas previamente detectados se hace necesario validar el producto y la calidad del mismo

Luego de analizar las posibles complicaciones que influyen en la calidad del producto final, se convierte en una necesidad el desarrollar un correcto proceso de pruebas para depurar las posibles fallas entre lo que debe ser y lo que se ha realizado. Por lo que el **problema a resolver** consiste en: ¿Cómo garantizar la disminución de defectos de la aplicación Sistema para Manejo Integral de Perforación de Pozos (SIPP)?

Para comprobar la calidad del sistema se propone realizar un adecuado diseño de pruebas, que una vez aplicado permita conocer los errores sobre los cuales trabajar, para lograr estabilidad y eficiencia del software. Por ende el **objeto de estudio de la investigación** es: El proceso de pruebas como parte del desarrollo de software; enmarcando el **campo de acción**: Aplicación de las pruebas de software al Sistema para Manejo Integral de Perforación de Pozos.

Basado en lo anteriormente mencionado el **objetivo general** de este trabajo es: Diseñar y aplicar casos de prueba al Sistema para Manejo Integral de de Pozos que garantice la disminución de defectos, tiempo y esfuerzo de trabajo por parte del equipo de desarrollo. Concluyendo como **hipótesis** que: Si se diseñan y aplican casos de prueba efectivos al Sistema para Manejo Integral de Perforación de Pozos entonces se podrá minimizar esfuerzo y tiempo del proceso de desarrollo del software.

Para el cumplimiento del objetivo planteado se trazaron **tareas de investigación**, las cuales permitirán organizar y llevar adelante la investigación:

1. Caracterizar las metodologías y tendencias existentes que permitan establecer el estado del arte en el desarrollo de pruebas de software, los principales conceptos y herramientas relacionados con la calidad del software.
2. Definir el enfoque de prueba y las configuraciones del entorno de prueba de acuerdo a las características del proceso productivo de la UCI y del producto.
3. Definir los casos de prueba para la estructuración de la ejecución de las pruebas.
4. Aplicar las pruebas al producto SIPP del proyecto Sistema para Manejo Integral de Perforación de Pozos.
5. Evaluar los resultados obtenidos de la aplicación de las pruebas.

Una vez concluida la investigación se espera obtener los siguientes **resultados**:

- Plan de Pruebas.
- Diseño de Casos de Pruebas.
- Evaluación de Pruebas.

Durante la realización de la investigación, se utilizarán varios métodos científicos:

Métodos teóricos: Permiten estudiar las características del objeto de investigación que no son observables directamente. Facilitan la construcción de modelos e hipótesis de investigación. (7).

Los siguientes métodos permiten estudiar las características de los procesos de prueba de un software que no son observadas directamente.

- **Método histórico-lógico**: Para investigar la información que se tiene hasta el momento sobre las pruebas y resumir los aspectos fundamentales necesarios para la investigación en curso.
- **Método analítico sintético**: Para la segmentación de los aspectos obtenidos sobre el desarrollo de pruebas para un mejor estudio y análisis de las características de las mismas dentro del flujo de desarrollo del software.
- **Método de modelación**: Este método se utiliza para modelar estrategias, diseñar los casos de prueba, pues es una necesidad que se ejecute esta modelación cuando se aplicará una prueba.

Según Hernández de León los **métodos empíricos:** Crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad. Posibilitan el conocimiento del estado del arte del fenómeno, su evolución en una etapa determinada, su relación con otros fenómenos. (7) Los métodos empíricos describen y explican las características del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional.

- **Observación:** Es el método que se utiliza para interactuar de forma directa o indirecta con el sistema, puede ser una observación de tipo externa incluida, debido a que se forma parte del grupo observador y se participa en él, mientras dura la investigación.
- **Observación bibliográfica**: Es el método utilizado en toda la investigación pues se recoge información de los aspectos tratados en la tesis desde el punto de vista de otros autores así como sus definiciones y resultados para permitir realizar un análisis del arte en ese aspecto.
- **Entrevista:** Con este método se aspira obtener un conocimiento cualitativo del sistema, se utilizará la entrevista no estructurada, que es más abierta, se trata el tema en cuestión sin necesidad de un cuestionario, es decir solo obtener información de forma espontánea y manejable.

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

Capítulo 1. Fundamentación teórica: En este capítulo se resume y analiza los contenidos fundamentales relacionados con las pruebas de software y sus elementos fundamentales, las metodologías de desarrollo, incluyendo las principales técnicas y métodos de prueba a utilizar en el desarrollo de la investigación.

Capítulo 2. Planificación, Diseño y Aplicación de las pruebas: En el presente se hará referencia al sistema en el que trabajamos, al desarrollo y seguimiento de las principales actividades como son la planificación de las pruebas, la configuración del entorno donde serán realizadas, los procedimientos a utilizar para un buen diseño y ejecución de los casos de prueba, para lograr un proceso definido y correctamente estructurado que se enmarque en los resultados que se deben obtener del producto.

Capítulo 3. Registro y Evaluación de los resultados: Se realizará una valoración de los resultados obtenidos luego de la aplicación de las pruebas al software, identificando y resumiendo los principales errores encontrados durante las pruebas para la posterior corrección y depuración de estos.

Capítulo 1:
Fundamentación Teórica.

Introducción

Una de las etapas más importantes del ciclo de desarrollo de un software es la fase de pruebas, pues con la misma se busca corregir y perfeccionar el producto desde sus inicios hasta la entrega del mismo al cliente, logrando que este cumpla exitosamente con todas sus funcionalidades y esté libre de defectos. En este capítulo se abordan los temas fundamentales con los que se trabajará a lo largo de la investigación, referidos principalmente a la ingeniería de pruebas, aspectos relacionados con la calidad de software, conceptos y definiciones expuestos por varios autores, los principales elementos de las pruebas de software, destacando algunos criterios generales, de los cuales se presentará su definición.

1.1 Calidad del software. Conceptos y definiciones.

La calidad del software es un aspecto importante que se le dedica mucho esfuerzo en la construcción de un producto, pero este casi nunca obtiene la perfección.

Según varios autores la calidad es:

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

“El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas”. (5)

En fin se exponen varios conceptos de calidad expresadas por personalidades y normas que rigen la calidad a nivel mundial.

“La calidad debe ser especificada, planificada, administrada, medida y certificada. Esto implica una visión integral que arroja la comprobación del software con el fin de lograr un mayor grado de satisfacción y confianza del cliente hacia la organización productora de software. Constituye entonces las pruebas de software, tarea de alta prioridad para las empresas productoras” (4).

1.1.1 Factores que miden la calidad del software.

Es importante conocer del tema de la calidad del software y los factores que determinan la misma. Estos se enfocan en 3 aspectos importantes de un producto de software:

- **Operaciones del producto**: Características operativas.
- **Revisión del producto:** Capacidad para soportar cambios.
- **Transición del producto**: Adaptabilidad a nuevos entornos.

1.2 Pruebas de Software.

Las pruebas de software constituyen un factor fundamental para la garantía de la calidad del software, representan una revisión final del diseño, las especificaciones y codificación del software. “Durante la fase de inicio puede hacerse parte de la planificación inicial de las pruebas, es decir, cuando se define el ámbito del sistema, sin embargo, la realización de las pruebas se centra en las fases de elaboración y transición para probar lo construido como resultado de la implementación, este proceso consiste en ejercitar el programa con la intención específica de descubrir errores previos a la entrega al usuario.”(14)

Es una actividad en la cual un sistema o componente es ejecutado bajo diferentes condiciones o requerimientos específicos, donde los resultados son observados, registrados y se realiza una evaluación de algún aspecto del sistema o componente. Forman un conjunto de herramientas, técnicas y métodos que permiten verificar y revelar la calidad de un producto de software.(8)

Efectuar pruebas a un sistema informático no significa necesariamente que el proceso de desarrollo esté asegurado y tampoco que de manera directa esté mejorando. Pero implementar un proceso de pruebas de software y más aún sostenerlo en tiempo, es un buen inicio para más adelante aumentar el alcance y los hallazgos registrados en su producto. Obteniendo la calidad requerida en el software a través de las pruebas, se logra reducir el número de errores, se alcanza una mayor fiabilidad para las funciones que debe realizar el mismo, mayor eficiencia e integridad de los datos, así como mayor flexibilidad y reusabilidad.(14)

1.2.1 Objetivos de las pruebas de software.

Dentro de los objetivos fundamentales que se persiguen al aplicarle las pruebas a un software se encuentran los siguientes:

- Brindar un mayor nivel de confiabilidad en los productos que se van generando.
- Detectar fallas o errores.
- Aumentar la calidad del producto final.

Los objetivos anteriores cambian la idea que normalmente se tiene cuando se plantea que una prueba exitosa es aquella donde no se detectan errores. El objetivo principal es diseñar pruebas que sistemáticamente reflejen diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. (1)

1.2.2 Estrategia de Pruebas.

Una estrategia de prueba del software integra las el proceso de prueba en una serie de pasos bien planificados que llevan a la construcción correcta del software. Una estrategia de prueba para el software debe constar de pruebas de bajo nivel, así como de pruebas de alto nivel.

Las características generales son:

- La prueba comienza en el nivel de módulo y trabaja “hacia afuera”.
- En diferentes puntos son adecuadas a la vez diferentes técnicas de prueba.
- La prueba y la depuración son actividades diferentes.(1)

Más concretamente, los objetivos de la estrategia de prueba son:

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de unidad, integración y las pruebas de sistema. Las pruebas de unidad y de integración son necesarias dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.
- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, cómo realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas.
- Realizar diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Los productos de desarrollo de software en los que se detectan defectos son probadas de nuevo y posiblemente devueltas a otra etapa, como diseño o implementación, de forma que los defectos puedan ser arreglados.(14)

Para conseguir estos objetivos el flujo de trabajo de la etapa de pruebas consta de las siguientes etapas:

- Planificación de las pruebas.
- Diseño de las pruebas.
- Implementación de las pruebas.
- Ejecución de las pruebas.
- Evaluación de las pruebas.(14)

Los responsables de realizar las actividades y los productos de desarrollo del software son:

Diseñador de pruebas: Es responsable de la planificación, diseño, implementación y evaluación de las pruebas. Esto conlleva generar el plan de pruebas y modelo de pruebas, implementar los casos de prueba y evaluar los resultados de las pruebas. Los diseñadores de prueba realmente no llevan a cabo las pruebas, sino que se dedican a la preparación y evaluación de las mismas.

Probador: Es responsable de desarrollar las pruebas de unidad, integración y sistema, lo que incluye ejecutar las pruebas, evaluar su ejecución, recuperar los errores y garantizar los resultados de las pruebas. (4)

En la presente investigación las funciones de los roles mencionados son llevados a cabo por la misma persona, debido a que tienen gran relación en la organización y ejecución de las tareas.

1.2.3 Plan de Pruebas.

Plan de pruebas describe la estrategia, recursos y planificación de las pruebas. La estrategia de prueba incluye la definición del tipo de pruebas a realizar para cada iteración y sus objetivos, el nivel de cobertura de prueba y el porcentaje de prueba que deberían ejecutarse con un resultado específico. (Ver Anexo 1)

El Plan de pruebas proporciona la siguiente información:

- La definición de los objetivos de las pruebas en el ámbito de la iteración.
- La definición de los elementos que se van a probar.
- Una explicación del enfoque o estrategia que se usará.
- Los recursos y planificación necesarios.
- Los resultados que se obtienen del proceso de prueba.(11)

También se define el plan estratégico que se va a seguir en las pruebas. Es necesaria una estrategia de pruebas para convencer al gestor y a otros stakeholders de que el enfoque es sensato y alcanzable. Una estrategia de prueba debe contener la siguiente información:

- Una explicación del enfoque general que se usará.
- Especificar tipos, técnicas y estilos de prueba que se emplearán como parte de la estrategia para cada una de las pruebas

1.2.4 Diseño Casos de Pruebas.

Los diseños de casos de prueba se utilizan para verificar que los componentes del sistema interaccionan entre sí de la forma apropiada, ya que se diseñan un conjunto de casos de prueba que permitan alcanzar los objetivos trazados en el plan de prueba con un esfuerzo mínimo. Para lograr lo anteriormente expresado se examinan un conjunto de casos de prueba con un solapamiento mínimo, cada uno de los cuales prueba un camino o escenario interesante a través de una realización de casos de uso.

Los propósitos de diseñar las pruebas son:

- Identificar y describir casos de pruebas para cada construcción.
- Identificar y estructurar los procedimientos de prueba especificando como realizar los casos de prueba.(14)

1.2.4.1 Casos de Prueba.

El propósito de un caso de prueba es especificar una forma de probar el sistema, incluyendo las entradas con las que se prueba, los resultados esperados y las condiciones bajo las que ha de probarse. Los casos de prueba son un producto de desarrollo de software, que ayudan a validar y verificar las expectativas de los stakeholders.

Los requerimientos son la fuente principal para obtener los casos de prueba pero no son el único medio y muchas veces son insuficientes para proporcionar una base completa que permita desarrollar las pruebas. Por lo que es necesario considerar otros elementos como riesgos, restricciones, tecnologías, cambios, fallos, etc. Normalmente, un Caso de Prueba se deriva de un caso de uso en el modelo de casos de uso. Con estos Casos de Prueba se validan los requerimientos funcionales del sistema. (14)Los siguientes son Casos de Prueba comunes:

- Un Caso de Prueba que especifica cómo probar un caso de uso o un escenario específico de un caso de uso. Un Caso de Prueba de este tipo incluye la verificación del resultado de la interacción entre los actores y el sistema, que se satisfacen las precondiciones y pos condiciones especificadas por el caso de uso y que se sigue la secuencia de acciones especificadas por el caso de uso.
- Un Caso de Prueba que especifica cómo probar una realización de caso de uso-diseño o un escenario específico de la realización. Un Caso de Prueba de este tipo puede incluir la verificación de la interacción entre los componentes que implementan dicho caso de uso.

Se debería desarrollar un Caso de Prueba para cada escenario del caso de uso. Los escenarios de un caso de uso se identifican describiendo los distintos caminos del flujo básico y flujo alternativo del caso de uso. (4)

1.2.5 Evaluación de Pruebas.

El resumen de evaluación de prueba organiza y presenta un análisis de los resultados de la prueba y las medidas clave de la prueba para su revisión y valoración, habitualmente por parte de los implicados. Además, el resumen de evaluación de prueba puede contener una sentencia general de calidad relativa y proporciona recomendaciones para esfuerzos de prueba futuros. El propósito de la actividad, es evaluar los resultados (medidas cuantificables) de las pruebas para determinar la calidad del producto desarrollado y la calidad del proceso de pruebas. (14)

Los diseñadores de pruebas llevan a cabo esta actividad revisando y evaluando los resultados de las pruebas, para lo cual comparan los resultados obtenidos con los objetivos esbozados en el plan de prueba.

Basándose en el análisis de la tendencia de los defectos los diseñadores de prueba pueden sugerir otras acciones, como ejemplo:

- Realizar pruebas adicionales para localizar más defectos, si la fiabilidad medida sugiere que el sistema no está suficientemente maduro.
- Relajar el criterio para las pruebas, si los objetivos de calidad para la iteración actual se pusieron demasiados altos.
- Aislar las partes del sistema que parecen tener una calidad aceptable y entregarlas como resultado de la iteración actual. Las partes que no cumplieron los criterios de calidad han de ser revisadas y probadas de nuevo. Los diseñadores de pruebas documentan la compleción de la prueba, su fiabilidad y sugieren acciones en una descripción de la evaluación de la prueba.

1.3 Elementos fundamentales del proceso de prueba.

1.3.1 Niveles de Pruebas.

La prueba es aplicada con diferentes objetivos y en disímiles escenarios o niveles de prueba, los que deben fluir de forma orgánica y ascendente, se considera que deben realizarse todos para garantizar la calidad de forma continua e incremental. Se distinguen los siguientes niveles de pruebas:

1.3.1.1 Prueba de Unidad.

Se centra en el proceso de verificación en la menor unidad del diseño del software: el componente o módulo. (1). Es la prueba enfocada a los elementos testeables más pequeño del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca. (15)

1.3.1.2 Prueba de Integración.

El objetivo de las pruebas de integración es verificar el correcto ensamblaje entre los distintos componentes una vez que han sido probados unitariamente con el fin de comprobar que interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales especificados en las verificaciones correspondientes. La prueba de integración contiene dos tipos fundamentales Integración incremental y no incremental; la primera de ellas combina los componentes y se incrementa progresivamente, mientras que la segunda se prueban los componentes por separado y luego se integran. (1)

1.3.1.3 Prueba de Sistema.

Las pruebas de sistema tienen como objetivo ejercitar profundamente el sistema comprobando la integración de la información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica. La prueba de sistema se integra por disímiles pruebas que se encargan de comprobar características específicas del software, contiene la prueba de Recuperación que verifica la recuperación del sistema ante fallos, la prueba de Seguridad que se encarga de confirmar los mecanismos de control de acceso al sistema, la prueba de Resistencia y Rendimiento que como su nombre lo indica comprueban el funcionamiento ante cargas masivas y el tiempo de respuesta del sistema respectivamente. También contiene pruebas que se enfocan en la validación de funciones, asistencia al usuario como son las pruebas de Funcionalidad y Usabilidad. La prueba de sistema recoge mayor cantidad de pruebas, ya que debe comprobar la mayoría de las funciones y aspectos del software.

1.3.1.4 Prueba de Aceptación.

El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Contiene una prueba tipo de prueba llamada Prueba de Alfa y Beta que se lleva a cabo los desarrolladores para descubrir errores que parezca que solo el usuario final puede descubrir. (11)

1.3.1.5 Niveles de pruebas seleccionados.

Luego de ser caracterizados, evaluados los diferentes niveles de prueba y revisado el entorno de aplicación, se selecciona la Prueba de **Unidad**, pues se busca la detección de errores de cada módulo por individual, centrando el proceso de verificación en la menor unidad del diseño del software. Se decide utilizar el nivel de **Integración** porque es necesario comprobar cómo funciona el intercambio de información en cada una de las interfaces expuestas por los módulos y su interacción a partir de las funcionalidades una vez que se hayan integrado los programas o componentes previamente probados. Se concluye con la prueba de **Sistema** porque se hace necesario verificar el programa final, luego que todos los módulos del sistema han sido integrados, asegurando que el sistema funciona en conjunto cabalmente.

1.3.2 Tipos de Pruebas de software.

Cuando se aplican pruebas a un software se hace necesaria la utilización de varias técnicas que facilitan comprobar el funcionamiento del mismo de una forma más detallada. Existen diferentes técnicas de prueba de software, las cuales se nombran a continuación:

- Pruebas de Funcionalidad.
- Pruebas de Usabilidad.
- Pruebas de Fiabilidad.
- Pruebas de Rendimiento.
- Pruebas de Soportabilidad.

1.3.2.1 Pruebas de Funcionalidad.

La prueba de funcionalidad se enfoca en requerimientos para verificar que se corresponden directamente a casos de uso o funciones y reglas del negocio. Esta técnica de prueba se basa en el método de caja negra y consiste en verificar la aplicación y sus procesos interactuando con la aplicación por medio de la interfaz de usuario y analizar los resultados obtenidos.

- **Prueba Funcional:** La prueba funcional tiene como objetivo asegurar el trabajo apropiado de los requisitos funcionales; incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.
- **Prueba de Seguridad:** La prueba de seguridad y control de acceso se enfoca en dos áreas de seguridad: seguridad en el ámbito de aplicación, incluyendo el acceso a los datos y a las funciones de negocios y seguridad en el ámbito de sistema, incluyendo conexión, o acceso remoto al sistema.
- **Prueba de Volumen:** La prueba de volumen somete al software a grandes cantidades de datos para determinar si se alcanzan límites que causen la falla del software. La prueba de volumen identifica la carga máxima continua que puede manejar el software a prueba en un período dado.

1.3.2.2 Pruebas de Usabilidad.

La prueba de usabilidad consiste en realizar pruebas efectuadas con usuarios, con el objetivo de determinar si la organización de los contenidos y las funcionalidades que se ofrecen desde el sitio web son entendidas y utilizadas por los usuarios de manera simple y directa. Las pruebas tradicionales son la de Boceto Web que tiene como objetivo entender la navegación, verificar si se pueden cumplir tareas y si el usuario entiende todos los elementos que se le ofrecen (8).La Prueba de Interfaz de Usuario verifica que la interfaz del usuario proporcione al usuario el acceso y navegación a través de las funciones apropiadas. (11)

1.3.2.3 Pruebas de Fiabilidad.

La fiabilidad de un software es la probabilidad de que un programa realice su objetivo satisfactoriamente, sin fallos, en un determinado periodo de tiempo y en un entorno concreto. Las pruebas de fiabilidad están constituidas por prueba de integridad, prueba de estructura, prueba de stress y prueba de falla y recuperación. La prueba de Fiabilidad contiene pruebas como la prueba de integridad, Estructura, Estrés, Falla y Recuperación: encaminadas a comprobar la robustez y eficiencia del software. (11)

1.3.2.4 Pruebas de Soportabilidad

Las pruebas de soportabilidad son otras de las pruebas que se le pueden realizar al software para solucionar problemas de una aplicación. Existen diferentes tipos de pruebas de soportabilidad, las cuales son: prueba de configuración y prueba de instalación que su principal fundamento es verificar el funcionamiento del sistema con diferentes configuraciones y que funcione bajo diferentes condiciones, una vez instalado el software. (11)

1.3.2.5 Tipos de pruebas seleccionadas.

Luego de analizar las características de los tipos de prueba, se decide utilizar la prueba funcional, prueba de seguridad y prueba de interfaz de usuario. La prueba funcional enfoca su trabajo en el funcionamiento de los requisitos funcionales, entrada de datos y navegación, mientras que la prueba de seguridad permite verificar el control del acceso a los datos y al sistema según el usuario que accede, finalmente la prueba de interfaz de usuario permite comprobar el acceso y navegación a través de las funciones apropiadas.

1.3.3. Métodos de prueba de software.

Los métodos de prueba de software tienen como objetivo diseñar pruebas que descubran diferentes tipos de errores con menor tiempo y esfuerzo. Proporcionan un mecanismo de ayuda para asegurar la calidad con la que cuenta un software y la mayor probabilidad de descubrimiento de errores que tiene el mismo. Los métodos tradicionales son:

- Método de Caja Negra.
- Método de Caja Blanca.

1.3.3.1 Método de Caja Negra.

El método de caja negra se centra en los requisitos funcionales y se lleva a cabo sobre la interfaz del software. Tiene como objetivo demostrar que las funciones del software son operativas, que las entradas se acepten de forma adecuada y se produzca un resultado correcto, teniendo en cuenta

que la integridad de la información externa se mantenga.. A continuación se citan diversas técnicas de dicho método.

- **Técnica de prueba basada en grafos:** En esta técnica se debe entender los objetos que se modelan en el software y las relaciones que conectan a estos, tales como objetos de datos, objetos de programa como módulos o colecciones de sentencias del lenguaje de programación. (10)
- **Partición equivalente:** La técnica de prueba partición equivalente divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. En otras palabras, este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia. De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente, a una prueba realizada con cualquier otro valor de dicha clase.
- **Análisis de Valores Límite:** El análisis de valores límite (AVL) es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase. (8)
- **Prueba de la tabla ortogonal**: La prueba de la tabla ortogonal puede aplicarse a problemas en que el dominio de entrada es relativamente pequeño pero demasiado grande para posibilitar pruebas exhaustivas.
- **Adivinando el error:** La idea básica es enumerar una lista de errores posibles o de situaciones propensas a error y después escribir los casos de prueba basados en la lista. (10)

1.3.3.2 Método de Caja Blanca.

El método de caja blanca denominada a veces prueba de caja de cristal, requiere del conocimiento de la estructura interna del programa. Permite comprobar los caminos lógicos del software y examinar el estado del programa en varios puntos para determinar si el estado es real y coincide con el esperado. Algunas de las técnicas empleadas en las pruebas de caja blanca son los siguientes:

- **La prueba del camino básico:** La prueba del camino básico 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.

- **Prueba de ruteo**: La prueba de ruteo tiene como objetivo ejecutar al menos una vez cada sentencia. Para ejecutar esta prueba se necesitan varios casos de prueba, entre estas está determinar varios caminos independientes y cada condición tomada debe cumplirse en un caso pero en el otro no.
- **Prueba de condición:** Método que ejercita las condiciones lógicas contenidas en un módulo del programa. Una condición simple es una variable booleana o una expresión relacional. Esta prueba se concentra en la prueba de cada condición del programa para asegurar que no contiene errores.
- **Prueba de bucle:** La prueba de bucle se concentra exclusivamente en la validez de la construcción de bucles. Se pueden definir cuatro tipos de bucles: simples, anidados, concatenados, no estructurados. (11)

1.3.3.3 Métodos de prueba seleccionados.

Luego de analizar los métodos y sus características, se decide que el más conveniente para las pruebas es el método de caja negra, pues se enfoca en los requisitos funcionales y se ejecuta sobre la interfaz del sistema, permitiendo comprobar las funcionalidades del programa. Específicamente se utilizará el método de partición equivalente ya que se dirige a la definición de casos de prueba que descubren clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar y se enfoca en los casos de uso.

1.4 Metodologías de desarrollo de software.

Para obtener un software con una óptima calidad, es necesario lograr un correcto ciclo de desarrollo de software, que permita organizar y efectuar satisfactoriamente los procesos que intervienen en la construcción del mismo. La correcta realización de estos procesos implica la utilización de metodologías de desarrollo que guíen la forma en que se aplica la ingeniería de software elevando la productividad del ciclo de desarrollo en aras de lograr un producto confiable y eficiente. A continuación se valoran algunas metodologías.

1.4.1 Metodología SCRUM.

SCRUM es aplicable en cualquier proyecto en el que exista una lista de funcionalidades o bloques de trabajo por realizar, un entorno complejo con requisitos cambiantes y un equipo de desarrollo asignado a dicha tarea. Se basa en un enfoque iterativo, donde cada iteración se denomina Sprint, al final de cada Sprint se obtiene un producto entregable, priorizándose aquellos aspectos que aportan mayor funcionalidad y valor al dueño del producto. Es una metodología especialmente indicada para pequeños equipos de desarrollo y se orienta a una entrega rápida de resultados. (16)

1.4.2 Metodología Extreme Programing (XP).

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo, corto equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. La metodología se basa en

- **Pruebas Unitarias:** se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro.
- **Re fabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares:** consiste en que cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.(10)

1.4.3 Metodología Proceso Unificado de Modelado (RUP).

Dentro de las metodologías fuertes la que más se destaca es el Proceso Unificado de Modelado (RUP). Este es uno de los procesos más generales que existe, está enfocado a cualquier tipo de proyecto así no sea de software, se basa en la documentación generada en cada uno de sus cuatro fases: Inicio (puesta en marcha), 2. Elaboración (definición, análisis y diseño), 3. Construcción (implementación) y 4. Transición (fin del proyecto y puesta en producción) en las cuales se ejecutarán varias iteraciones (según el tamaño del proyecto). (1)

1.4.3.1 Flujo de Trabajo de Pruebas de RUP.

La metodología RUP describe cómo planear y ejecutar las pruebas teniendo en cuenta un grupo de artefactos, actividades y trabajadores. RUP define sus artefactos según roles y actividades, los artefactos son resultados tangibles producto de las actividades que se realizan, mientras que las actividades constituyen la guía para realizar cada uno de los procesos, sin embargo no se debe obviar a los trabajadores que realizan las actividades de cada flujo y obtienen los artefactos.

Artefactos del flujo de trabajo de prueba.

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

- **Artefacto Plan de prueba:** Este artefacto define los objetivos de las pruebas en el ámbito de la iteración (o el proyecto) los elementos de destino, el enfoque que se adopta, los recursos necesarios y los entregables que se deben generar.

- **Artefacto 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.
- **Artefacto 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. Un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular, por ejemplo, ejercitar un camino concreto de un programa o verificar el cumplimiento de un determinado requisito.
- **Artefacto 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. Este artefacto resume el análisis de uno o más registros de prueba y solicitudes de cambio, proporcionando una valoración relativamente detallada de la calidad de los elementos de destino de la prueba y el estado del esfuerzo de prueba. (13)

  En el caso de CALIDAD UCI se utilizan los siguientes artefactos:
  - Artefacto Plan de prueba.
  - Artefacto caso de prueba.
  - Artefacto defecto o Informe No conformidades.

  En la presente investigación se utilizan los siguientes artefactos**:**
  - Artefacto Plan de Pruebas.
  - Artefacto Diseño de Casos de Pruebas.
  - Artefacto Evaluación de Pruebas.

Trabajadores del flujo de trabajo de prueba.

- **Jefe de Pruebas:** Es el responsable del éxito de la prueba, Este rol es el defensor de 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.
- **Diseñador de pruebas:** Planifica las pruebas. Además son los responsables de la evaluación de las pruebas de integración y de sistema cuando éstas se ejecuten. 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 la mismas. Es el responsable durante las actividades principales de las pruebas, el cual incluye la conducción de las pruebas necesarias y el registro del resultado de la prueba.(13)

En la presente investigación, se implementará los roles:

- Diseñador de Pruebas.
- Probador.

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:** En esta actividad se deben identificar y describir todas las pruebas que se van a realizar para garantizar la calidad del producto desarrollado. El propósito de la actividad es planificar los esfuerzos de prueba de una iteración siguiendo las siguientes actividades, describiendo una estrategia de prueba, estimando los requisitos para el esfuerzo de prueba y planificando el esfuerzo de la prueba. (14)
- **Diseñar las pruebas**: Esta actividad permite estructurar y organizar bien las pruebas. El diseñador de pruebas analiza los objetivos de las pruebas que se han planificado y desarrolla los casos de prueba y el modelo de pruebas necesario para llevarlas a cabo.
- **Implementar prueba:** El propósito de esta actividad es implementar los casos de prueba que se han definido en la actividad anterior de diseño de las pruebas, automatizando todos los que sean posibles. Los componentes de prueba, clases de prueba y datos de prueba se crean usando los casos de prueba como entrada.
- **Evaluar pruebas:** El propósito de la actividad de pruebas es evaluar los resultados (medidas cuantificables) de las pruebas para determinar la calidad del producto desarrollado y la calidad del proceso de pruebas. Los diseñadores de pruebas llevan a cabo esta actividad revisando y evaluando los resultados de las pruebas, para lo cual comparan los resultados obtenidos con los objetivos esbozados en el plan de prueba. (14)

1.4.4 Metodología de desarrollo de software seleccionada.

En la presente investigación se utiliza la metodología RUP, teniendo en cuenta sus características y ventajas. Dicha metodología proporciona un eficiente ciclo de desarrollo en la fabricación de un software, permite realizar aportes y resultados tangibles. En el presente trabajo se profundiza en el flujo de trabajo de prueba, sus roles, actividades y artefactos que se generan como resultado. Finalmente se debe mencionar como importante característica, que RUP se adapta a las características de la empresa y mide los recursos involucrados en el proceso.

Conclusiones Parciales

En el presente capítulo se realiza un análisis de la información de los principales temas necesarios para el desarrollo de la investigación. Se evalúan los conceptos fundamentales de calidad de software, pruebas de software y elementos bases que la conforman. Se caracterizan los enfoques principales para el diseño de casos de pruebas, sus conceptos y métodos. Los contenidos abordados durante el capítulo son indispensables para el análisis, comprensión y desarrollo del proceso de diseño de pruebas.

Capítulo 2:
Planificación, Diseño y Aplicación de las pruebas.

Introducción

En el capítulo anterior se trataron los principales conceptos relacionados con la investigación que constituyen una guía para el presente capítulo, donde se describe la estrategia de trabajo llevada a cabo en el proceso de pruebas, donde inicialmente se describe el sistema a probar; se elabora el plan de pruebas que recoge los recursos necesarios para el entorno de prueba , especificando cuáles fueron los niveles, técnicas y métodos que se aplicaron; así como una breve descripción de cómo se fueron empleando cada uno de ellos.

2.1 Sistema para Manejo Integral de Perforación de Pozos.

El sistema SIPP se encarga de gestionar el flujo de información de los diversos procesos que intervienen en la perforación petrolera, registrando y analizando las variables de dichos procesos. Es una aplicación web que termina su primera versión con un conjunto de funcionalidades totalmente operativas, está conformado por 6 subsistemas y cada uno divide en una serie de módulos. Estos subsistemas contienen información que es diversa pero se relacionan entre sí, a continuación los subsistemas: Seguridad, Pozo, DIPP, CEINPET, Visualización de Información y Manejo de Archivo.

A continuación se muestra la relación de los subsistemas con sus respectivos módulos y casos de uso. Cada caso de uso tiene una prioridad crítica por lo tanto es necesario un caso de prueba para cada uno de ellos.

- **Subsistema Seguridad**
  - ✓ Módulo Inicio.
    - ✓ CU_Autenticar Usuario.
    - ✓ CU_Cambiar Contraseña.
  - ✓ Módulo Administración.
    - ✓ CU_Gestionar Usuario.
    - ✓ CU_Gestionar Rol.
    - ✓ CU_Gestionar Permiso.
- **Subsistema Pozos**
  - ✓ CU_Gestionar Reporte Diario de Perforación.
  - ✓ CU_Gestionar Reporte Diario Operativo.
  - ✓ CU_Gestionar Cronograma de Perforación Planificado.
  - ✓ CU_Gestionar Perforación Diaria.

- ✓ CU_Generar Reporte de Récord de Barrenas.
- ✓ CU_Gestionar Reporte Composición de Herramientas.
- ✓ CU_Gestionar Reporte de Propiedades de Fluido de Perforación.

- **Subsistema DIPP**
  - ✓ CU_Gestionar Información Inicial del Pozo.
- **Subsistema CEINPET**
  - ✓ CU_ Generar Parte Diario de Geología.
  - ✓ CU_ Gestionar Columna Litológica de Pozo.
  - ✓ CU_ Construir Columna Litológica de Pozo.
  - ✓ CU_ Gestionar Litología.

- **Subsistema Visualización de Información**
  - ✓ CU_ Generar Gráficas de Reportes.
  - ✓ CU_ Visualizar Información de Reportes.
  - ✓ CU_ Buscar Información General.
  - ✓ CU_ Buscar Información Específica.

- **Subsistema de Manejo de Archivos**
  - ✓ CU_Gestionar Trabajo con Archivo.

2.2 Características a examinar para las pruebas.

En el software se deben verificar inicialmente ciertos detalles, desde que se tiene contacto con ella, observar, si funciona bien y sin errores cuando se ejecuta, si las funcionalidades se ejecutan correctamente y permite una fácil interacción con el cliente.

No obstante tiene gran importancia adentrarse en las particularidades del sistema, como la correspondencia de las funcionalidades con sus respectivos casos de uso y requerimientos. Para lograr el buen funcionamiento del producto de software, los requisitos funcionales deben ser verificados y probados en el sistema; ya que constituyen el comportamiento del mismo y muestran cómo los casos de uso son llevados a la práctica. Para verificar lo anteriormente mencionado se hace necesario la planificación y ejecución de un correcto proceso de pruebas que permita entregar el software con una mejor calidad.

2.3 Estrategia de Prueba.

La estrategia de prueba constituye una guía de las pruebas al sistema, refleja el conjunto de acciones que se realizan en dicho proceso de manera organizada y secuencial. La estrategia busca el objetivo de evaluar activamente el producto, comenzando por los elementos más simples y avanzando progresivamente hasta probar todo el software en su conjunto.

En la presente investigación la estrategia de prueba comprende un grupo de actividades que a continuación se nombran:

- Realización del plan de prueba.
- Definición de los niveles, técnicas y método de prueba a utilizar.
- Realización de pruebas exploratorias al software.
- Realización de diseños de casos de prueba.
- Realización de pruebas funcionales al software.
- Elaboración de la plantilla de no conformidades.
- Evaluación de los resultados.

2.3.1 Realización del Plan de prueba.

La realización del Plan de Prueba es un factor principal para la puesta en marcha de un buen proceso de prueba, en el mismo se plantean un conjuntos de pasos y actividades que indican cómo se realizará dicho proceso. En dicho informe se tuvieron en cuenta las características del software, el equipo de trabajo, los recursos del sistema necesarios para llevar a la práctica las pruebas, los posibles escenarios de los casos de pruebas, también se evidencian los requerimientos a probar y los diferentes técnicas y métodos necesarios para evaluación. (Ver Anexo 1)

Respecto al orden de pruebas, una práctica frecuente es la siguiente:

1. Se define el alcance de las pruebas y el entorno de trabajo, para la posterior definición de los trabajadores, los recursos de hardware y software que se necesitan
2. Se planifican los posibles escenarios de pruebas y los elementos fundamentales de pruebas a utilizar.
3. Pasar pruebas de caja negra analizando valores límite, donde se ven las condiciones límite de entrada y de salida.
4. Identificar clases de equivalencia de datos (entrada y salida) y añadir más pruebas de caja negra para contemplar valores normales (en las clases de equivalencia en que estos sean diferentes de los valores límite; es decir, en rangos amplios de valores).

2.3.2 Recursos del Sistema.

Para la ejecución de las pruebas es necesario contar con un conjunto de recursos que sean compatibles con la aplicación. (Ver Anexo 1)

Especificaciones de Software.

SO	Servidor WEB	Servidor Base Datos
Windows XP, GNU/Linux, Mac.	Apache	PostgreSQL 8.2

Especificaciones de Hardware.

Servidores	SO	Procesador	RAM	HDD	Tarjeta Red
Servidor profesional	Server 2003	Dos Micro Core 2 Duo 2.4 GHZ	4 GB	Límite indefinido.	No definido
Servidor Pozo	XP Profesional	Celeron D 3.0 GHZ	512 MB	80 GB	Inalámbrica

Tabla 1: Requisitos del Sistema.

2.3.3 Definición de los niveles, técnicas y método de prueba a utilizar.

- **Niveles de Prueba seleccionados.**

  Luego del estudio de los distintos niveles de prueba realizado en el capítulo anterior se deciden utilizar las pruebas de Unidad, Integración y Sistema. Se decide utilizar la prueba de unidad ya que evalúa cada módulo por separado, el cumplimiento de cada uno de sus requisitos y casos de uso. Conjuntamente verifica cada funcionalidad y sus respectivas interfaces.

  Se decide seleccionar la prueba de integración pues es necesario comprobar el intercambio de información en las interfaces expuestas por los módulos del sistema y su interacción a partir de estas funcionalidades, una vez que se hayan integrado los módulos y subsistemas de la aplicación.

  Se concluye con la prueba de sistema pues verifica el programa final, luego de integrar los componentes de software y hardware, asegurando que la incorporación del software a los elementos del sistema funciona correctamente y realiza las debidas funciones.

- **Tipos de Pruebas seleccionados**.

  Después del estudio realizado sobre los diversos tipos de prueba y sus características, se llega a la conclusión de que las más factibles para el proceso son: la prueba funcional, prueba de seguridad, prueba de interfaz de usuario.

  Se aplicará la prueba funcional ya que asegura el funcionamiento apropiado del objeto de prueba, incluyendo la navegación, entrada de datos, proceso de errores y recuperación. La prueba de

seguridad pues permite determinar los niveles de permiso de usuarios, las operaciones de acceso al sistema y acceso a datos. Por último la prueba de interfaz de usuario porque verifica a través de la navegación la estandarización de las interfaces y su vinculación teniendo en cuenta el usuario autenticado, también las pruebas de carga y estrés porque es necesario comprobar el rendimiento del sistema soportando la cantidad máxima de usuarios conectados y su comportamiento al aumentar esta carga con los mismos recursos disponibles.

- **Métodos de Prueba seleccionados.**

Teniendo en cuenta el estudio realizado de los métodos de pruebas se decide utilizar el método de caja negra porque es una buena alternativa para comprobar las funcionalidades del sistema centrándose en los requisitos que exige el usuario final. Dentro de este método de prueba se utilizará la técnica de partición de equivalencia ya que se dirige a la definición de casos de prueba que descubren clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

2.3.4 Requerimientos Generales.

A continuación se especifican una serie de requisitos que enmarcan la funcionalidad del software, son a modo general, extraídos de cada módulo, para dar una idea de la funcionalidad del sistema y constituyen los elementos a probar:

- **Subsistema Seguridad:** El trabajo de este subsistema consiste en controlar el acceso a la aplicación, teniendo en cuenta los roles de cada uno de los que acceden y las funcionalidades que utilizan según la labor que desempeñan y donde.

Módulo Inicio:

CU_ Autenticar Usuario: Se encarga de la seguridad del sistema; es el encargado de proporcionar el acceso inicial al sistema solo a aquellos usuarios registrados.

CU_ Cambiar Contraseña: Se encarga de que algún usuario de los que pueden interactuar con la aplicación solicite cambiar su contraseña de acceso a la misma.

Módulo Administración.

CU_ Gestionar Usuario: Se encarga de la seguridad del sistema; tiene como funcionalidad el trabajo con los datos de los usuarios que autorizados a trabajar con el sistema.

- **Subsistema DIPP**

  CU_ Generar Parte Diario Operativo de Perforación: Este caso de uso se encarga de generar automáticamente el Parte Diario de Perforación, sin necesidad de que el usuario inserte ningún dato.

- **Subsistema Pozo**

  CU_ Gestionar Inventario Barrena: El caso de uso se inicia cuando se necesita insertar o actualizar datos del Inventario de Barrena.

  CU_ Generar Reporte Diario Operativo: Este caso de uso se encarga de generar automáticamente el Reporte Operativo del Pozo, sin necesidad de que el usuario inserte ningún dato.

  CU_ Generar Reporte de Récord de Barrenas: Este caso de Uso se encarga de Generar el Reporte de Récord de Barrenas.

2.3.5 Realización de pruebas exploratorias.

Las pruebas exploratorias son la primera etapa del proceso de pruebas y se realizaron con el objetivo de explotar las vulnerabilidades mínimas del sistema, donde los requerimientos son menos precisos y más fluidos. Partiendo de la estrategia de trabajo trazada, se exploró la aplicación prácticamente a ciegas y sin empleo de la documentación. Durante su ejecución se intentó comprender cómo funciona la aplicación y producir condiciones de error. Se intentó además obtener información de los desarrolladores y deducir la interacción entre los componentes.

2.3.6 Diseño de Casos de Prueba.

Conociendo que un caso de prueba se deriva de un caso de uso del sistema, se diseñó un caso de prueba para los casos de uso seleccionados. Con estos casos de prueba se verificaron los requerimientos funcionales en el que se especificó cómo probar un caso de uso o un escenario específico del mismo, para verificar el resultado de la interacción entre el usuario final y el sistema, para conocer si se satisfacían las precondiciones y pos condiciones especificadas. Para detallar cada caso de prueba se utilizó una tabla que consta de los siguientes campos:

- Nombre de la sección: se especificó el nombre de la sección [SC 1: Nombre de la sección].
- Escenarios de la sección: se especificaron los escenarios de la sección [EC 1.1: Nombre del Escenario].
- Descripción de la funcionalidad: se describió brevemente la funcionalidad.
- Flujo central: se describieron los pasos a desarrollar para probar la funcionalidad que se indicó.

A partir de las descripciones del caso de uso donde se especificaban los distintos caminos del flujo básico y flujo alterno, se realizó una descripción de las variables que se encontraban en todas las interfaces y que están asociadas al caso de uso, al cual se le está realizando el caso de prueba.

En esta descripción se llenaron algunos campos por los cuales está conformada dicha tabla:

- No: se enumeró todos los campos o variable, descrito en el caso de uso.
- Nombre de campo: se especificó el nombre del campo de entrada.
- Clasificación: se especificó la clasificación según el componente de diseño utilizado [ejemplo: campo de texto, lista desplegable o combo box, entre otros].
- Puede ser nulo: se especificó si el campo puede ser nulo o no,
- Descripción: se describieron brevemente los datos que debían introducirse.

Dando lugar a que se realizara un juego de datos utilizando las clases válidas y no válidas, identificadas con la técnica partición equivalente, con el objetivo de probar la validez de cada uno de los datos que se introdujeron en el sistema a través de sus entradas. En la plantilla de diseño de casos de prueba se llenaron los siguientes campos para cada sección del caso de uso:

- Id del escenario: se especificó el id del escenario [EC1, EC2,…,ECn]
- Escenario: se especificó el nombre del escenario.
- Variables [1, 2,…, n]: se especificó el nombre de la variable y en su celda correspondiente se indicó el valor del dato [V (Válido), I (Inválido), N/A (No Aplica)].
- Respuesta del sistema: se escribió el resultado que se esperaba al realizar la prueba.
- Resultado de la prueba: se escribió el resultado que se obtuvo al realizar la prueba. (8)

2.3.6.1 Casos de Pruebas.

Subsistema Seguridad

Caso de Uso: Autenticar Usuario.

Descripción General: El caso de uso es uno de los encargados de la seguridad del sistema; proporciona el acceso inicial al sistema, sólo a aquellos usuarios registrados.

Condiciones de Ejecución**:** Se muestra un formulario solicitando usuario y contraseña.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**
SC 1: Autenticar Usuario.	EC 1.1: Autenticar Usuario correctamente.	Luego de introducir los datos necesarios para autenticarse, el sistema comprueba en la base de datos que la información sea correcta, consulta el rol y los permisos de este para el acceso a las funcionalidades correspondientes.

	EC 1.2: Autenticar con datos incorrectos.	El sistema comprueba los datos, al ser incorrectos muestra un mensaje de error informando que los datos de autenticación son erróneos.
	EC 1.3: Autenticar dejando el campo de usuario vacio.	El sistema comprueba los datos, el campo de usuario no puede estar vacío y por tanto lanza un mensaje de error

Tabla 2: Secciones a probar CU_ Autenticar Usuario.

Caso de Uso: Cambiar Contraseña.

Descripción General: El caso de uso se inicia cuando algún Usuario de los que pueden interactuar con la aplicación solicita cambiar su contraseña de acceso a la misma.

Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Cambiar Contraseña	EC 1.1: Cambiar contraseña satisfactoriamente.	Luego de acceder a la aplicación, se opta por cambiar la contraseña, se debe completar campos con la antigua y luego la nueva al aceptar la opción, el sistema verifica que la contraseña antigua esté correcta y realiza el cambio satisfactoriamente.
	EC 1.2: Cambiar contraseña con datos erróneos.	Luego de acceder a la aplicación, se opta por cambiar la contraseña, se añade la contraseña antigua incorrectamente el sistema verifica que el dato es erróneo y emite un mensaje notificando el error.
	EC 1.3: Cambiar contraseña dejando un campo vacio.	Luego de acceder a la aplicación, se opta por cambiar la contraseña, dejando uno de los campos vacios, el sistema emite un mensaje notificando la falta de la información necesaria para realizar el cambio.

Tabla 3: Secciones a probar CU_ Cambiar Contraseña.

Caso de Uso: Gestionar Usuario.

Descripción General: El caso de uso es uno de los encargados de la seguridad del sistema; tiene como funcionalidad el trabajo con los datos de los usuarios que autorizados a trabajar con el sistema.

Condiciones de Ejecución: El actor debe estar autenticado como administrador del sistema.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Insertar Datos del Usuario	EC 1.1: Insertar Usuario correctamente.	El sistema muestra un formulario con campos necesarios para registrar al usuario. Se comprueban que los datos estén correctos, los campos completados con los tipos de datos especificados y se agrega satisfactoriamente al usuario.
	EC 1.2: Insertar Usuario con datos incorrectos.	El sistema comprueba los datos insertados, al ser incorrectos muestra un mensaje de error informando que los datos del usuario son incorrectos.
	EC 1.3: Insertar Usuario dejándolos campos obligatorios vacíos	El sistema comprueba los datos y los campos que no estén completados, luego muestra un mensaje de error avisando que se deben completar.
SC 2:Actualizar Datos del Usuario	EC 2.1: Seleccionar Usuario a modificar.	El sistema muestra una lista desplegable que permite seleccionar el usuario que se desea modificar, luego muestra los datos de este.
	EC 2.2: No seleccionar usuario a modificar	El sistema muestra una lista desplegable que permite seleccionar el usuario que se desea modificar, al no seleccionar ninguno debe mostrar un mensaje de error.
SC 3 : Listar/Eliminar Usuario	EC 3.1: Listar / Eliminar el usuario satisfactoriamente.	El sistema muestra la lista de los usuarios que están en el sistema y se selecciona el deseado a eliminar.
	EC 3.2: Listar / Eliminar el usuario dejando campos vacíos.	El sistema muestra la lista de los usuarios, si no se selecciona ningún campo debe mostrar un error y no cumple con la funcionalidad.

Tabla 4: Secciones a probar CU_Gestionar Usuario.

- **Subsistema DIPP**

Caso de uso: Generar Parte Diario Operativo de Perforación.

Descripción General: Este caso de uso se encarga de generar automáticamente el Parte Diario de Perforación, sin necesidad de que el usuario inserte ningún dato.

Condiciones de Ejecución: Reporte Diario Operativo del Pozo. El actor debe estar autenticado como administrador del sistema.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1.1 :Generar Parte Diario Operativo de Perforación	EC: 1.1 Genere y muestre correctamente el parte.	El sistema muestra un formulario donde se solicita la fecha y el parte que se desea, recopila los datos registrados anteriormente para generar el parte y luego lo muestra sin posibilidad de realizar ningún cambio.
	EC: 1.2 Seleccionar los criterios de generación del parte incorrectamente o nulos.	En el sistema muestra un formulario, donde se solicite la fecha y el parte que se desea, en este caso puede resultar algún error en la selección y no muestra el parte.

Tabla 5: Secciones a probar CU_Generar Parte Diario de Perforación.

- **Subsistema Pozo.**

Caso de Uso: Generar Reporte Diario Operativo.

Descripción General: Este caso de uso se encarga de generar automáticamente el Reporte Operativo del Pozo, sin necesidad de que el usuario inserte ningún dato.

Condiciones de Ejecución: Reporte Diario de Perforación. El actor debe estar autenticado previamente en el sistema.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Generar Reporte Diario Operativo	EC: 1.1 Generar el reporte satisfactoriamente.	El sistema muestra un formulario de selección de fecha como criterio de búsqueda del reporte, luego consulta los datos registrados anteriormente y muestra el reporte.
	EC: 1.2 No muestra el reporte por falta de datos.	El sistema muestra un formulario de selección de fecha como criterio de búsqueda del reporte, cuando consulta los datos, no lo tiene, por tanto no muestra el reporte.
	EC: 1.3 Insertar una fecha fuera de rango.	El sistema al consultar la selección de la fecha debe mostrar los datos de esta, pero en este caso no debe proceder y debe mostrar error o ningún dato.

Tabla 6: Secciones a probar CU_Generar Reporte Diario Operativo.

Caso de Uso: Generar Reporte de Récord de Barrenas.

Descripción General: Este caso de Uso se encarga de Generar el Reporte de Récord de Barrenas

Condiciones de Ejecución: Que el usuario este autenticado previamente en el sistema.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Generar Reporte de Récord de Barrenas	EC: 1.1 Generar el reporte satisfactoriamente.	El sistema recopila los datos del reporte y visualiza una lista de barrenas con sus respectivos datos.
	EC: 1.2 No se genera el reporte por falta de datos.	El sistema no tiene registrados datos suficientes para confeccionar y mostrar el reporte

Tabla 7: Secciones a probar CU_Generar Reporte Récord de Barrenas.

Caso de Uso: Gestionar Inventario Barrena.

Descripción General: El caso de uso se inicia cuando se necesita insertar o actualizar datos del inventario de Barrena.

Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Insertar Datos en el Inventario Barrena	EC: 1.1 Insertar los datos en el inventario satisfactoriamente.	El sistema muestra un formulario para insertar los datos necesarios para registrar el inventario, luego comprueba que la información sea correcta y con el tipo de dato correspondiente, inserta los datos y muestra un mensaje de éxito de la operación.
	EC: 1.1 Insertar los datos en el inventario y que estos no sean correctos.	El sistema muestra un formulario para insertar los datos necesarios, al comprobar si son correctos y los tipos de datos correspondientes, encuentra que no lo son y emite un mensaje de error.
	EC: 1.2 Insertar los datos en el inventario dejando campos vacíos o nulos.	El sistema muestra el formulario para insertar los datos necesarios, al comprobar los campos, existen alguno de estos vacíos, por tanto el sistema emite un mensaje de error y cancela la operación.

SC 2: Modificar Datos en el Inventario Barrena	EC: 2.1 Modificar los datos en el inventario correctamente.	El sistema muestra un listado de todas las barrenas con sus datos correspondientes, permite que estos sean seleccionados y modificados, luego muestra los cambios y emite un mensaje de éxito de la operación.
	EC: 2.2 Modificar los datos y que estos no se muestren.	El sistema muestra un listado de todas las barrenas con sus datos correspondientes, permite que estos sean seleccionados y modificados, en el caso de ser incorrectos el sistema no lo muestra y visualiza un mensaje de error.
	EC: 2.3 Modificar los datos dejando sus campos vacíos o nulos.	El sistema muestra un listado de todas las barrenas con sus datos correspondientes, permite que estos sean seleccionados y modificados, si al modificar se dejan campos vacíos el sistema lanza una advertencia y no realiza la operación.

Tabla 8: Secciones a probar CU_Gestionar Inventario Barrena.

Conclusiones Parciales

Durante el desarrollo del presente capítulo se describió el proceso de pruebas del Sistema de Manejo Integral de Perforación de Pozos, basado en la estrategia elaborada. Se realizaron pruebas a la aplicación, para ello se aplicaron las técnicas y métodos, en los niveles seleccionados. Simultáneamente se elaboraron los principales artefactos resultantes del proceso como el Plan de Pruebas y los diseños de los casos de prueba a partir de los casos de uso del sistema. Los resultados arrojados de todas las pruebas realizadas se archivaron en la plantilla de no conformidades, para darle seguimiento y garantizar que se fueran eliminando a medida que avanza el proceso de pruebas.

Capítulo 3
Registro y Evaluación de los resultados.

Introducción.

En el presente capítulo se llevará a cabo un levantamiento y evaluación de los resultados obtenidos en la aplicación de las pruebas al sistema. Una de las principales tareas que se tuvo en cuenta, por su importancia para elevar la calidad del producto, fue llevar el registro de los errores encontrados en la aplicación de los casos de pruebas y en las pruebas funcionales realizadas a la aplicación. Para validar el proceso realizado se hará un análisis de los resultados obtenidos en cada una de las pruebas contra los resultados esperados, aspecto primordial que se tuvo en cuenta en la detección de no conformidades.

3.1 Análisis y Evaluación de los casos de Pruebas.

Durante la realización y aplicación de los casos de pruebas, surge la interrogante de ¿Cuán efectivos pueden ser los casos de pruebas?, para responder a la duda planteada se propone efectuar una evaluación de los mismos mediante las listas de chequeo, dichas listas pueden ser aplicadas al sistema directamente o a los casos de pruebas. En la investigación se aplicaron a los diseños de casos de prueba, como función básica para su valoración, las mismas contribuirán a conocer cuál es el valor de los diseños y cuanta seguridad y confianza brinda.

Las listas de chequeo verifican varios aspectos de los diseños, mediante preguntas y evaluaciones, principalmente se valora:

1) Estructura del documento.
2) Elementos definidos por la metodología.
    a) Diseño de la prueba
3) Realización de la prueba.
4) Semántica del documento.

En la investigación se realizaron un total de 7 casos de prueba, durante la aplicación de las listas de chequeo a dichos casos de prueba, se encontraron fallas menores específicamente en el Índice y Reglas de confidencialidad del documento, sin embargo las listas arrojaron una evaluación de Bien para los casos de pruebas en su totalidad.(Ver Anexo 2).

3.2 Realización de pruebas funcionales al software.

Durante la realización de las pruebas funcionales al software se analizaron los distintos valores válidos e inválidos que pueden tomar las variables relacionadas con los casos de uso y con cada una de las funcionalidades. Se comenzó a probar por los niveles definidos, aplicando los tipos de pruebas previamente seleccionados. Para continuar el procedimiento de todo lo que se deseaba verificar en cada uno de los niveles de prueba utilizados, se explotaron diversas técnicas que permitieron realizar una excelente revisión, se puede nombrar a la prueba funcional, prueba de interfaz de usuario y prueba de seguridad.

- **Prueba de Unidad e Integración.**

  En este nivel de prueba se comprobó la funcionalidad y validez de todas las entradas de datos en cada módulo, con empleo de los diseños de casos de prueba**.** Se verificó el flujo de datos de los módulos. Se encontraron errores como funcionalidades incorrectas o inexistentes, defectos referentes a las interfaces, errores en estructuras de datos o en accesos a bases de datos, así como inconsistencias con respecto a lo especificado en los requisitos funcionales.

- **Prueba de Sistema.**

  Por último se llevaron a cabo las pruebas de sistema para verificar la funcionalidad y el rendimiento total del sistema, en este caso se realizaron pruebas de seguridad, de control de acceso a datos y funcionalidad. Luego de que el intercambio de información entre los módulos fuese eficiente, se analizó el software como un todo, comprobando que todos los módulos funcionan correctamente de forma íntegra. Para probar estos aspectos en el software se usaron como entradas un conjunto de datos de pruebas para llevar a cabo el procesamiento de las instrucciones y de esta manera examinar los resultados en los demás módulos, comprobando que existiera correspondencia entre ellos según lo ejecutado anteriormente.

- **Prueba Funcional.**

  La prueba funcional se aplicó teniendo en cuenta los casos de uso, se ejecutó cada flujo de caso de uso o función, usando juegos de datos válidos y no válidos, para verificar aspectos como:

  - ✓ Los resultados del sistema cuando se emplearon datos válidos fueron los esperados.
  - ✓ El uso de datos no válidos o funciones desplegaron mensajes de error o advertencia apropiados.

- **Prueba interfaz de usuario.**

Para la aplicación de las pruebas de interfaz de usuario se realizó un manejo de ventanas y métodos de acceso a interfaces, para verificar de esta manera las características, tamaño, posición, estado, de acuerdo a los estándares de cada uno de los campos. A través de un conjunto de pantallas se comprobó la facilidad de navegación de las mismas y su seguimiento según los estándares establecidos. Se verificó además mediante la navegación por todos los casos de uso que las interfaces fueran sencillas, se comprendieran fácilmente y se correspondieran con el tipo de usuario que estuviera autenticado en ese momento.

- **Prueba de seguridad.**

La prueba de seguridad se ejecutó teniendo en cuenta las probabilidades que tenía el usuario de acceder al sistema, ya sea manera correcta o ilegal. Las principales aspecto que se evaluaron respondieron de forma satisfactoria.

- ✓ Para probó la seguridad en el ámbito de aplicación, se identificó cada tipo de usuario y se hizo una lista con las funciones y los datos sobre las que cada uno de ellos tenía permiso.
- ✓ Se verificó le permiso de cada tipo de usuario ejecutando operaciones específicas para cada uno de ellos.
- ✓ Se modificó el tipo de usuario y se ejecutaron las operaciones del tipo anterior para el usuario modificado. En cada caso según la modificación, se verificó que las funciones o datos adicionales estuvieran o no disponibles para el nuevo tipo de usuario en el sistema.

3.3 Análisis de los resultados obtenidos en la aplicación de las pruebas.

Posteriormente de aplicar los casos de pruebas, se realizó un análisis de las principales fallas encontradas, dentro de las cuales se encontraron errores en la documentación utilizada, específicamente en la correspondencia de esta con la aplicación. La revisión del sistema se realizó de forma minuciosa para encontrar la mayor cantidad de errores y evitar un incorrecto funcionamiento en las interfaces del sistema y en el software en general que en parte imposibilitarían una precisa comprensión y entendimiento del sistema.

Luego de haber obtenido los errores en cada caso de prueba se compararon los resultados que se esperaban al realizar las pruebas y los que verdaderamente se obtuvieron. Los resultados de las pruebas que no fueron satisfactorios se manifestaron como no

conformidades y se emitieron en el registro de efectos y dificultades detectadas que se encuentra en la parte final de cada diseño de caso de prueba con los siguientes campos:

- ✓ Elemento: se especificó el nombre del elemento.
- ✓ No: se especificó el número de la no conformidad.
- ✓ No conformidad: se describió la no conformidad.
- ✓ Aspecto correspondiente: se especificó el aspecto correspondiente a la no conformidad.
- ✓ Etapa de detección: se especificó la etapa de detección del error.
- ✓ Significativa: se puso una (X), en caso de que la no conformidad estuviera clasificada como significativa.
- ✓ No significativa: se puso una (X), en caso de que la no conformidad estuviera clasificada como no significativa.
- ✓ Recomendación: se puso una (X), en caso de que la no conformidad solo fuera una recomendación.
- ✓ Estado NC: se colocó el estado de la no conformidad y la fecha, cada vez que se revisó se dejó el estado anterior y se colocó 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, es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica el por qué. (8)

Los principales defectos detectados se deben a los resultados en las entradas de datos, los errores de interfaz y funcionalidades que aun no están implementadas y son necesarias para el funcionamiento correcto de otras. La mayoría de los diseños de casos de prueba aplicados a cada caso de uso de los diferentes módulos con los que cuenta el sistema, arrojaron no conformidades, que quedaron plasmadas de una manera organizada, o sea, cada caso de prueba que se aplicó consta de una tabla propia de él, con las no conformidades encontradas y su clasificación las cuales se resumen a continuación. (Ver Anexos)

Subsistemas	Significativas	No significativas	Recomendaciones
Seguridad	10	1	11
DIPP	3	0	3
Pozo	6	0	5
TOTAL	19	1	19

Tabla 9: Datos de Defectos.

Después de haber detallado cada una de las no conformidades por módulos se puede concluir diciendo que se obtuvieron un total de 21 no conformidades, clasificadas en 20 significativas y 1 no significativas; y se obtuvo un total de 19 recomendaciones. Presentando mayor cantidad defectos el subsistema de Seguridad con 11 No conformidades, de ellas 10 significativas.

La aplicación en general estaba bien implementada, la mayoría de las funcionalidades que requería el sistema cumplían su objetivo. Las no conformidades se clasifican en Aplicación y Documentación. Las no conformidades más importantes que arrojaron los diseños de casos de prueba fueron:

Dentro del tipo de Aplicación:

- ✓ Correspondencia con documentación.
- ✓ Validación.
- ✓ Ortografía
- ✓ Funcionalidad.
- ✓ Excepciones.

Dentro del tipo Documentación:

- ✓ Ortografía.
- ✓ Redacción.
- ✓ Error Técnico.

Para determinar la fiabilidad del sistema se crearon gráficas de las tendencias de los defectos, donde se representan el porcentaje y distribución de los errores según los tipos específicos de defectos. A continuación se representa como se comportaron las no conformidades del tipo de Aplicación, donde se apreció que la más detectada fue la de correspondencia con la documentación y la de validación, debido a que la mayoría de los casos de uso, no expresan fielmente las respuestas del sistema y no proporcionan los posibles flujos alternos que puedan ocurrir. En el caso de la validación el sistema expresó que presenta grandes problemas de validación, principalmente en los casos de entrada de caracteres no válidos.

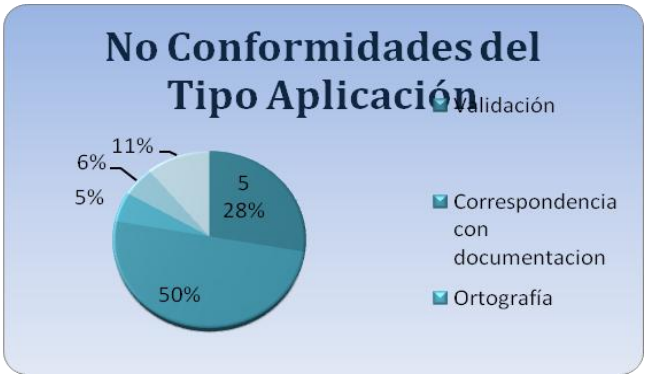


Figure 1: No Conformidades del Tipo Aplicación.

Luego de haber analizado las no conformidades del tipo aplicación de puede mostrar que existen errores en la documentación con que se trabajó y esto influyó en la cantidad de errores detectados. El objetivo de la investigación no enmarca revisar la documentación pero detectaron las siguientes no conformidades del tipo Documentación. En la siguiente gráfica se representa:

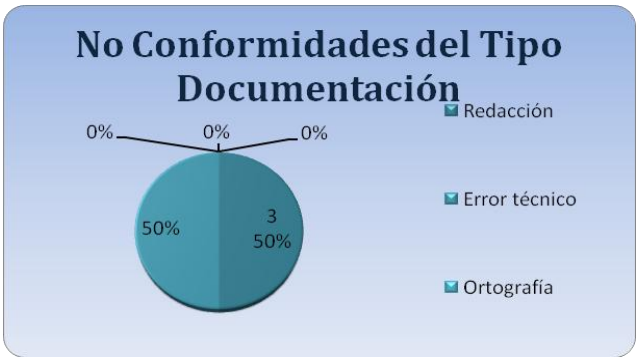


Figure 2: No Conformidades del Tipo Documentación.

La aplicación en general estaba bien implementada, la mayoría de las funcionalidades que requería el sistema cumplían su objetivo. Las no conformidades más importantes que arrojaron los diseños de casos de prueba fueron:

- ✓ Interfaces que no está correctamente validadas.
- ✓ Falta de Datos cuando se necesita generar reportes.
- ✓ Funcionalidades que no están bien descritas en sus respectivos casos de uso.
- ✓ Información incorrecta en la guía de usuarios con respecto a lo que realmente brindaba la interfaz.

En cuanto a la interfaz, los principales errores clasificados como no significativos, estuvieron dados por faltas de ortografía que presenta en todos sus módulos, incluyendo los mensajes mostrados por el sistema. No son errores de una importancia alta que conlleven a una incorrecta ejecución del programa o a un mal funcionamiento del mismo, pero que sí afectan la calidad integral y la estética del software. La recomendación que más abundó estuvo dada en el mejoramiento de los enfoques de los mensajes y en las abreviaturas utilizadas para un mejor entendimiento y comprensión del usuario.

Una vez examinada la cantidad de errores encontrados de la aplicación de los casos de pruebas y la clasificación de las mismas, además de las recomendaciones que se le hicieron, se puede resumir diciendo que a pesar de no estar caracterizadas como catastróficas sí podían traer

como consecuencia una mala aceptación por parte del cliente. Por esta razón, se hizo mucho hincapié en los procedimientos que generarán fallas y no encontrarán las salidas esperadas o datos mostrados incorrectamente, los cuales podían atentar contra la confiabilidad y funcionalidad del sistema.

Conclusiones Parciales

En el capítulo se realizó una evaluación del proceso de pruebas, el cual mostró una serie de errores en el sistema, que permitió caracterizarlos y registrarlos. Estos resultados son de gran importancia a tener en cuenta por el equipo de desarrollo para próximas versiones del sistema, para darle seguimiento a los fallos encontrados y las posibles causas de los mismos, lo que facilitaría el tratamiento de errores y la prevención de defectos. Para evaluar la documentación utilizada, es decir (Casos de Pruebas) se utilizó las listas de chequeo (Ver Anexo 2) atendiendo a diversos parámetros de manera individual y así comprobar la efectividad de los diseños elaborados. Analizado esto se concluye que el SIPP es un software portable, seguro en términos de acceso a la información precisa, usable atendiendo a las características de su interfaz, eficiente, pero parcialmente confiable, dado los fallos que presentó y su insuficiente recuperación con respecto a estos.

Conclusiones Generales

El gran desarrollo de la industria del software ha incrementado la necesidad de que los productos realizados sean confiables y precisos, aumentando la exigencia por parte de clientes y usuarios finales en general. Por ende es necesario que los sistemas sean desarrollados con prácticas de Ingeniería de Software adecuadas y que en su etapa de comprobación hayan sido probados correctamente para lograr el nivel de calidad requerido. Es por ello que resulta de vital importancia que se tenga en cuenta como parte trascendental del proceso de desarrollo de software, velar por que la calidad de los mismos, esté a la altura del desarrollo que se exige en estos tiempos. Al planificarse y ejecutarse el proceso de pruebas desarrollado se obtuvieron los siguientes resultados:

- ✓ Se definió una estrategia de trabajo logrando disminuir los defectos, tiempo y esfuerzo del equipo de desarrollo.
- ✓ Se logró desarrollar una adecuada documentación de todo el procedimiento de pruebas como el Plan de pruebas que sirvió de apoyo para la realización del proceso.
- ✓ Se diseñaron los casos de prueba que cubrieron las funcionalidades del sistema, logrando con estos probar los requerimientos funcionales con los que cuenta el software.
- ✓ Se logró detectar una gran cantidad de errores que permiten mejorar el software y conocer las principales fallas y el origen de las mismas.
- ✓ Se analizaron los resultados de todas las pruebas realizadas dando una valoración de cada uno de ellos que sirvió para evaluar al software.

Con el estudio realizado y la aplicación de las pruebas se ha cumplido el objetivo propuesto, de desarrollar un proceso de pruebas eficaz al Sistema de Manejo Integral de Perforación de Pozos, con vista a la obtención del mayor número de faltas existentes en la misma. Permitiendo lograr con su posterior eliminación la obtención de una aplicación con el mínimo de errores y mayor calidad.

Recomendaciones

Al concluir la presente investigación y luego de haber cumplido con los objetivos previstos se recomienda al equipo de trabajo:

- Se fomente en los proyectos productivos la realización de las pruebas durante todo el ciclo de vida del software, de manera que se detecten el mayor número de errores previamente al proceso de pruebas.
- Incluir dentro del proceso de pruebas, las pruebas de Carga y Estrés automatizadas con la herramienta JMeter, dado que con las pruebas funcionales no se asegura un rendimiento y desempeño del sistema una vez desplegado en la entidad que lo solicita.
- Eliminar todos los defectos detectados durante la primera iteración del proceso de prueba realizado al sistema y que se encuentran recogidos en este documento, en las plantillas de diseño de casos de pruebas y registro de no conformidades.
- Realizar nuevamente un proceso de detección de defectos al sistema y que este proceso sea ejecutado por un equipo de prueba, donde estén bien definidos todos los roles (Administrador de pruebas, Analista de pruebas, Diseñador de Pruebas y Probador) para lograr un trabajo más efectivo y una mayor calidad en el producto.

Referencias Bibliográficas

1.**Pressman, R.S.** *Ingeniería del software. Un enfoque práctico.* 3era Edición : McGrawHill (1992), 1992.

2. *Monográfico Calidad del Software / Software de calidad.* **Novatica.** Enero-Febrero 1999, Número 137.

3. **Natalia Juristo, Ana M. Moreno, Sira Vegas.** *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* 2006.

4. **Marcelino Cabrera, Pedro Cano, Miguel Lastra y otros.** Guía Virtual del Proceso de Desarrollo del Software. (Pruebas). *Departamento de Lenguajes y Sistemas Informáticos.* [Online] http://lsi.ugr.es/~arroyo/inndoc/inicio.php.

5. **Lovelle, Juan Manuel Cueva.** *Calidad de Software.* Universidad Nacional de la Pampa: s.n., 1999.

6. **Johanna, Rojas and Barrios., Emilio.** Métodos de prueba de caja negra. *Universidad Distrital.* [Online] 2007. [Cited: 2 2, 2009.] http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

7. **Hernández, L. y otros.** *El paradigma cuantitativo de la investigación científica.* Ciudad de la Habana, Cuba. : Editorial Universitaria (Eduniv), 2003.

8. **Camejo, Onaysi Vasallo Artigas Yislén Dolores Ramírez.** *Proceso de Pruebas de Liberación al Sistema de Manejo de Datos de Ensayos Clínicos Cubano.* Ciudad de la Habana: Universidad de las Ciencias Informáticas., 2009.

10. **CALISOFT.** *Curso Introducción a las pruebas de software. .* 2009.

11. **Betancourt Rodríguez, Keyly and Rodríguez Martell, Frank.** *Diseño de pruebas de calidad para producto LIMS control de calidad del CIGB.* Ciudad de la Habana: Universidad de las Ciencias Informáticas., 2007.

12. **Norma ISO 8402.** *International Standars Organization.Norma Española.* [Online] http://www.aenor.es [ISO].

13. EVA Entorno de Aprendizaje Virtual. [Online] http://eva.uci.cu/mod/resource/view.php?id=14103.

14**Booch G, Rumbaugh, J. y Jacobson, I;** *"El Proceso Unificado de Desarrollo de Software"*Addison-Wesley2000

15. Flujo de Trabajo de Pruebas. **Software., Departamento de Ingeniería y Gestión de.** Ciudad de La Habana: UCI, 2009.

16. **Proyectalis.** Proyectalis. *Gestión de Proyectos.* [En línea] Proyectalis, 2010. [Citado el: 17 de 05 de 2010.] http://www.proyectalis.com/servicios/formacion/scrum/.

Bibliografía Consultada

1. **Pressman, R.S.** *Ingeniería del software. Un enfoque práctico.* 3era Edición: McGrawHill (1992), 1992.

2. *Monográfico Calidad del Software / Software de calidad.* **Novatica.** Enero-Febrero 1999, Número 137.

3. **Johanna, Rojas and Barrios., Emilio.** Métodos de prueba de caja negra. *Universidad Distrital.* [Online] 2007. [Cited: 2 2, 2009.]

4. **Hernández, L. y otros.** *El paradigma cuantitativo de la investigación científica.* Ciudad de la Habana, Cuba. : Editorial Universitaria (Eduniv)., 2005

5. **Camejo, Onaysi Vasallo Artigas Yislén Dolores Ramírez.** *Proceso de Pruebas de Liberación al Sistema de Manejo de Datos de Ensayos Clínicos Cubano.* Ciudad de la Habana: Universidad de las Ciencias Informáticas., 2009.

6. **CALISOFT.** *Curso Introducción a las pruebas de software. .* 2009.

7. **Betancourt Rodríguez, Keyly and Rodríguez Martell, Frank.** *Diseño de pruebas de calidad para producto LIMS control de calidad del CIGB.* Ciudad de la Habana: Universidad de las Ciencias Informáticas., 2007.

8.http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

9. EVA Entorno de Aprendizaje Virtual. [Online] http://eva.uci.cu/mod/resource/view.php?id=14103.

10.**Booch G, Rumbaugh, J. y Jacobson, I;***"El Proceso Unificado de Desarrollo de Software"*Addison-Wesley2000

11. *Flujo de Trabajo de Pruebas.* **Software., Departamento de Ingeniería y Gestión de.** Ciudad de La Habana: UCI, 2009

Anexos

Anexo 1: Plan de Prueba

Plan de Pruebas
Sistema de Manejo Integral de la Perforación de Pozos (SIPP)

Versión 1.1

1. Introducción.

Este plan de pruebas proporciona el marco dentro del cual el equipo de prueba desarrolla las pruebas trabajando con los recursos y la planificación dada. En dicho informe se tuvieron en cuenta las características del software, el equipo de trabajo, los recursos del sistema necesarios para llevar a la práctica las pruebas, los posibles escenarios de los casos de pruebas, también se evidencian los requerimientos a probar y los diferentes técnicas y métodos necesarios para evaluación.

1.1Alcance

El alcance de este plan de pruebas comprende la planificación para la ejecución de las pruebas de validación que permitan probar los requisitos del sistema y el funcionamiento de los casos de uso, utilizando diferentes tipos de pruebas como son las pruebas de sistema, específicamente pruebas funcionales para verificar si el sistema funciona como un todo.

1.1Definiciones, Acrónimos y Abreviaturas.

DIPP: Dirección de Intervención y Perforación de Pozos.
CUPET: Empresa Cuba Petróleo.
CEINPET: Centro de Investigaciones del Petróleo
MINBAS: Ministerio de la Industria Básica
Gb: GigaBytes.
Mb: MegaBytes.
RAM: Random Access Memory.

1.2Referencias

Código	Título
[1]	Roles y responsabilidades SIPP.doc
[2]	Especificación de Requerimientos de Software.doc
[3]	Modelo de Despliegue de SIPP.doc
[4]	Modelo Casos de Uso del Sistema.doc

2. Roles y responsabilidades

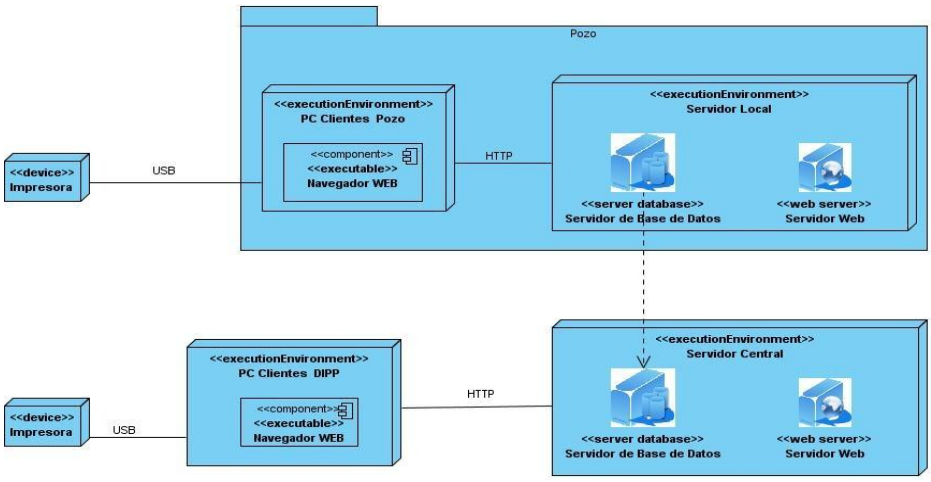
Rol	Cantidad	Responsabilidad
Diseñador de Pruebas	2	✓ Participa en la confección de la estrategia y el plan de pruebas. ✓ Identifica los métodos, las técnicas, herramientas y directrices apropiadas para implementar las pruebas necesarias. ✓ Encargado de diseñar casos de pruebas para el sistema. ✓ Dirige la definición del enfoque de pruebas y garantiza la implementación satisfactoria.

		✓ Diseña las pruebas. ✓ Genera los casos y datos de prueba. ✓ Puede fungir como probador. ✓ Analiza y evalúa el estado del producto de trabajo comprobado. ✓ Evalúa el desempeño del probador y la calidad de las pruebas realizadas
Probador	2	✓ Realiza las actividades principales de las pruebas, el cual incluye la guía de las pruebas necesarias y el registro del resultado de las pruebas. ✓ Ejecutar pruebas. ✓ Recuperar los errores. ✓ Documentar los defectos.

3. Escenario de pruebas.

- Evaluar todo lo entregado por los desarrolladores (Diseño de Casos de prueba; Especificación de requerimientos de software; Instalación de la aplicación; Manual de explotación; Diagramas.
- Comprobar que están listos todos los recursos humanos, materiales y/o financieros previstos en el “Plan de Pruebas”.
- Asignar las tareas al equipo de pruebas.
- Preparar los documentos de trabajo (rediseñar los casos de prueba si es necesario).

3.1Despliegue del sistema



3.2Recursos del sistema

3.2.1Servidores

Servidores	SO	Procesador	RAM	HDD
Servidor profesional	Server 2003	Dos Micro Corre 2 Dúo 2.4 GHZ	4 GB	Tamaño indefinido.
Servidor Pozo	XP Profesional	Celeron D 3.0 GHZ	512 MB	80 GB

3.2.2PC Clientes

PC Clientes	
Cantidad	3
Descripción	Equipos necesarios para la aplicación de las Pruebas.
Software base	• Sistema Operativo: Windows XP o Linux/Debían. • Servidor Web: Apache. • Servidor de Base de Datos: PostgreSQL. 8.2

4. Requerimientos a probar

Esta sección del Plan de Pruebas contiene una lista de todos los requisitos funcionales que serán probados. Cualquier requisito no incluido en esta lista estará fuera del alcance de las pruebas; Esto libera de responsabilidad al equipo de pruebas en caso de que existan problemas con la funcionalidad del sistema que esté relacionado con un requisito no incluido en este listado.

Subsistema Seguridad

Módulo Inicio.

CU_Autenticar Usuario.

CU_Cambiar Contraseña

Módulo Administración.

CU_Gestionar Usuario

CU_Gestionar Rol

CU_Gestionar Permiso.

Subsistema Pozos

CU_Gestionar Reporte Diario de Perforación.

CU_Gestionar Reporte Diario Operativo.

CU_Gestionar Cronograma de Perforación Planificado.

CU_Gestionar Perforación Diaria.

CU_Generar Reporte de Récord de Barrenas.

CU_Gestionar Reporte de Presupuesto Diario.

Subsistema DIPP

CU_Gestionar Información Inicial del Pozo.

Subsistema CEINPET

CU_ Generar Parte Diario de Geología.

CU_ Gestionar Columna Litológica de Pozo.

CU_ Construir Columna Litológica de Pozo.

CU_ Gestionar Litología.

Subsistema Visualización de Información

CU_ Generar Gráficas de Reportes.

CU_ Visualizar Información de Reportes.

CU_ Buscar Información General.

CU_ Buscar Información Específica.

Subsistema de Manejo de Archivos

CU_Gestionar Trabajo con Archivo.

5. Estrategia de pruebas de aceptación

Las pruebas que se realizarán son las conocidas como Pruebas de Caja Negra (funcionalidad), donde los casos de prueba se diseñarán considerando exclusivamente las entradas y salidas del subsistema, sin preocuparnos por la estructura interna del mismo.

5.1 Objetivo

El objetivo que se persigue con esta estrategia de pruebas es encontrar defectos en el software.

5.2 Técnica

Los casos de prueba serán desarrollados a partir de los casos de uso, al menos habrá un caso de prueba para cada caso de uso.

Se tomarán de referencia listas de chequeo para evaluar la interfaz de usuario, el diseño y atributos de calidad como son: portabilidad, eficiencia, confiabilidad y usabilidad de cada uno de los módulos a probar.

El diseño de las pruebas estará encaminado a la realización de pruebas de cubrimiento (invocar todas las funciones), pruebas de valores límites y clases de equivalencia de datos.

El diseño de las pruebas y los resultados de la misma, quedarán establecidos en el documento “Diseño de casos de prueba” y los resultados quedaran establecidos en el documento “No Conformidades”.

6. Evaluación de las pruebas

Las pruebas serán evaluadas a partir de criterios de aceptación respecto a las No Conformidades detectadas al final cada iteración de pruebas. Estos criterios para los defectos encontrados se clasifican en No Críticos y Críticos, donde los No Críticos pueden ser una simple desviación o incumplimiento de un elemento de un requisito y los Críticos son una desviación o incumplimiento de un requisito que afecta de manera significativa al sistema.

Se tendrá también una Lista de Chequeo dirigida a las pruebas para verificar si realmente

están bien elaborados los casos de pruebas.

7. Cronograma

No.	Tarea	Fecha	Responsable	Observaciones
1	Planificación de las pruebas	1/4/10-15/4/10.	Yisell Cruz Arias	Caracterizar los contenidos necesarios y planificar las pruebas enmarcando el producto y sus especificaciones.
2	Diseño de los casos de pruebas.	16/4/10-por definir	Yisell Cruz Arias	Identificar los casos de uso críticos y elaborar los casos de prueba aplicables según sus respectivas descripciones.
3	Ejecución de los casos de pruebas	Por definir	Yisell Cruz Arias	Aplicar los casos de pruebas al sistema y registrar los resultados de cada uno de ellas, tanto esperados como no esperados.
4	Evaluación de los resultados de las pruebas.	Por definir	Yisell Cruz Arias	Evaluar los resultados de las pruebas en cada una de las iteraciones para saber si se libera o no el producto.

Anexo 2: Lista de Chequeo

Lista de Chequeo Caso de Diseño de Caso de Prueba

Sistema de Manejo Integral de Perforación de Pozos SIPP

Versión 1.0

Introducción

La lista de chequeo pretende evaluar el contenido de los casos de prueba, antes de ser aplicados, teniendo en cuenta aspectos necesarios para que sean efectivos y tengan calidad, además de tener un punto donde se referencia las observaciones en cuanto al aspecto a evaluar.

Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del caso de prueba del proyecto Sistema de Manejo Integral de Perforación de Pozos.

Esta plantilla ha sido confeccionada para guiar al diseñador de pruebas y probador, en la verificación y evaluación de las especificaciones del caso de prueba. Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del [caso de prueba que se desarrollan.

Forma de Uso:

Peso: *Define si el indicador a evaluar es crítico o no.*

Evaluación (Eval): *Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.*

Cantidad de elementos afectados: *Especifica la cantidad de errores encontrados sobre el mismo indicador.*

Comentario: *Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.*

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.*

N.P. (No Procede): *Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.*

Estructura de la lista de chequeo Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?	0		0	
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)	0		0	
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Diseño de la prueba					
	1. ¿La descripción general del caso de prueba coincide con el resumen del Caso de Uso?	0		0	
crítico	2. ¿Se han especificado las condiciones iniciales que deben existir para que se realice el Caso de prueba?	0		0	
crítico	3. ¿Se ha especificado la sección o las secciones que tiene el Caso de Uso? Las secciones, son flujos excluyentes dentro del caso de uso. Ejemplo: en el caso de uso Gestionar Libro las secciones serían eliminar, modificar y registrar un nuevo libro.	0		0	
crítico	4. ¿Se han especificado todos los escenarios de pruebas por cada sección? Un escenario es un "camino" completo a través del caso de uso. El flujo básico seria el que cumple con el objetivo del caso de uso y los flujos alternos serian las	0		0	

	alternativas o excepciones.				
crítico	5. ¿Si el escenario tiene que ver con una entrada de datos, existe un escenario para validar que los datos sean correctos?	0		0	
crítico	6. ¿Si el escenario tiene que ver con una entrada de datos existe un escenario para verificar que no falten datos obligatorios?	0		0	
	7. ¿Aparece una descripción de la funcionalidad por cada escenario?	0		0	
crítico	8. ¿El flujo del escenario especifica claramente el camino para probarlo?	0		0	
crítico	9. ¿Si existen variables se han definido sus nombres?	0		0	
crítico	10. ¿Si existen variables se han definido su clasificación (de acuerdo al tipo de dato)?	0		0	
crítico	11. ¿Si existen variables se ha definido si aceptan valores nulos o no?	0		0	
	12. ¿Si existen variables se ha hecho una descripción de las mismas, especificando condiciones que debe cumplir?	0		0	
crítico	13. ¿Por cada escenario se han especificado las variables asociadas?	0		0	
crítico	14. ¿Por cada escenario se ha especificado el estado que las variable pueden tomar (V, I, NA)?	0		0	
crítico	15. ¿Si el escenario implica que las variables tomen valor inválido, se	0		0	

	han definido todas las posibles combinaciones, fijando una variable con el valor inválido y todas las demás con estado válido en cada combinación?				
crítico	16. ¿Se ha especificado la respuesta del sistema para cada escenario?	0		0	
crítico	17. ¿Durante la prueba se han colocado los valores que toman las variables en cada escenario?	0		0	
Realización de la prueba					
crítico	1. ¿Al realizar la prueba se ha especificado el resultado de la misma en cada escenario?	0		0	
crítico	2. ¿En el registro de defectos y dificultades detectados se especifica si el elemento revisado es documentación o aplicación?	0		0	
	3. ¿La descripción de la No Conformidad está descrita de forma clara y precisa?	0		0	
crítico	4. ¿El aspecto correspondiente a la No Conformidad especifica el camino a seguir para encontrar la No Conformidad?	0		0	
	5. ¿Se especifica la etapa en que se encontró la No Conformidad?	0		0	
crítico	6. ¿Se especifica el grado de importancia de la No Conformidad, (es significativa, no significativa) y en caso de ser una recomendación se especifica también?	0		0	
crítico	7. ¿Se especifica el estado de la No Conformidad indicando la fecha de cambio del estado?	0		0	
Semántica del documento					

Peso	Indicadores a Evaluar	Eval	(NP)	Cantidad de elementos afectados	Comentarios
Crítico	1. ¿No se han identificado errores ortográficos?	0		0	
Crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?	0		0	
	3. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?	0		0	
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?	0		0	

Evaluación del Artefacto
Evaluación: Bien

Nombre y Apellido del Evaluador: Yisell Cruz Arias

Anexo 3: Caso de Prueba Autenticar Usuario.

3.3 Descripción de Variables

SC 1: Autenticar Usuario.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Usuario	Campo de texto	No	No se pueden introducir (&, %, etc.) en ese campo.
2	Contraseña	Campo de texto	No	Puede introducirse letras y números y cualquier tipo de datos

3.4 Matriz de Datos.

SC 1: Autenticar Usuario.

Escenario	Usuario	Contraseña	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.1: Autenticar Usuario correctamente.	V (ycarias)	V (pruebasipp)	El sistema comprueba datos y permite el acceso satisfactoriamente finalizando el caso de uso.	*Satisfactorio*	El Sistema Verifica que los datos introducidos no contengan errores, habilita la opción Enviar, comprueba que los datos estén correctos, consulta la base de datos para chequear si el usuario existe. Verifica que la contraseña sea la correcta. Consulta el rol del usuario y los permisos del usuario. Muestra un mensaje de bienvenida a la aplicación.

Escenario	Usuario	Contraseña	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.2: Autenticarse erróneamente.	I (ycarias%%)	V (pruebasipp)	El sistema muestra un mensaje: “El usuario o la contraseña son incorrectos”, limpia los campos del usuario y la contraseña y actualiza la página.	*Satisfactorio*	El usuario inserta unos de los datos incorrectamente , de forma tal que cuando el sistema verifique la información en la base de datos la encuentre incorrecta, seguidamente El Sistema muestra un mensaje de que no se pueden introducir letras/números/caracteres(&,%, etc.) en ese campo, limpia todos los caracteres extraños y “El usuario o la contraseña son incorrectos”
	V (ycarias)	I (pruebasipp1)	El sistema muestra un mensaje: “El usuario o la contraseña son incorrectos”, limpia los campos del usuario y la contraseña y actualiza la página.	*Satisfactorio.*	
EC 1.3: Autenticar dejando el campo de usuario vacio.	I (vacío)	V (pruebasipp)	El sistema muestra un mensaje: “El usuario o la contraseña son incorrectos”, limpia los campos del usuario y la contraseña y actualiza la página.	*Satisfactorio.*	El usuario al insertar los datos, deja uno de los campos vacios, cuando el sistema envía la información, encuentra unos de los campos vacios y muestra un error. Usuario o Contraseña no válidos
	V (ycarias)	I (vacío)	El usuario accede sin problemas al sistema, debido a que no es un campo obligatorio la contraseña, al registrarse puede decidir no usar contraseña.	*No Satisfactorio*	

Anexo 4: Caso de Prueba Gestionar Usuario.

1. **Descripción de Variables.**

SC: Insertar Usuario.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	No	Puede introducirse letras y números. Es un campo obligatorio.
2	Usuario	Campo de texto	No	Puede introducirse letras, números y cualquier tipo de datos. Es un campo obligatorio.
3	Ubicación	Campo de selección	Si	Tiene 2 opciones seleccionar y consiste en radio button, por tanto siempre se selecciona uno
4	Roles Pozo	Campo de selección	No	Tiene 3 opciones y consiste en chekbox para añadir se debe seleccionar al menos 1 necesariamente. Es un campo obligatorio.
5	Pozo	Lista desplegable	Si	Lista desplegable con los pozos donde se puede trabajar.
6	Sexo	Lista desplegable	No	Lista desplegable con las opciones permitidas de los géneros. Es un campo obligatorio.
SC 2:Actualizar Datos del Usuario				
1	Usuarios	Lista desplegable	No	Lista desplegable con los usuarios registrados en el sistema.
SC: Listar/Eliminar Usuario.				
1	Nombre	Campo de Selección	No	Campo de selección con todos los usuarios y sus datos.

2. Matriz de Datos

SC 1: Insertar Usuario.

Escenario	Nombre	Usuario	Ubicación	Roles	Pozo	Sexo	Respuesta del Sistema	Resultado de Prueba	Flujo Central
EC 1.1: Insertar Usuario correctamente.	V Yisell	V ycarias	N/A	V 1..3	N/A	V (F)	El sistema comprueba datos e inserta el usuario satisfactoriamente.	*Satisfactorio.*	El Sistema muestra un formulario con los con los campos donde se va a insertar los datos del usuario, verifica si los datos son correctos. El Sistema muestra una ventana emergente solicitando confirmación de la operación, comprueba que los datos no contengan errores, almacena los datos en la base de datos y muestra un mensaje del éxito
EC 1.2: Insertar con datos incorrectos.	I Yisell%%	V ycarias 1	N/A	V 1..3	N/A	V Femenino	El sistema comprueba que la información insertada sea correcta si no es así lanza un mensaje de error.	No Satisfactorio	El Sistema muestra un formulario con los con los campos donde se va a insertar los datos del usuario, verifica si los datos son correctos en este caso no lo son, muestra el mensaje “Datos introducidos incorrectos” El Sistema muestra un mensaje de que no se pueden introducir letras/números/caracteres (&, %, etc.) en ese campo.

Escenario	Nombre	Usuario	Ubicación	Roles	Pozo	Sexo	Respuesta del Sistema	Resultado de Prueba	Flujo Central
	V Yisell1	I ycarias &&	N/A	V 1..3	N/A	V Femenino	El sistema comprueba que la información insertada sea correcta si no es así lanza un mensaje de error.	No Satisfactorio	El Sistema limpia todos los caracteres extraños. El Sistema no envía los datos y actualiza la página. El Sistema muestra un mensaje de error.
EC 1.3: Insertar Usuario dejando un campo vacio.	I (vacío)	V ycarias	N/A	V 1..3	N/A	V Femenino	El sistema verifica que los campos estén llenos , si no resulta , muestra un mensaje de error ” El campo NOMBRE DE USUARIO es	*Satisfactorio.*	El Sistema muestra un formulario con los con los campos donde se va a insertar los datos del usuario, verifica si los datos son correctos y que los campos obligatorios estén completados en este caso no lo están, muestra el mensaje “Los campos obligatorios no pueden estar vacios”. El Sistema limpia todos los caracteres extraños. El Sistema no envía los datos y actualiza la página.
	V Yisell	I (vacío)	N/A	V 1..3	N/A	V Femenino	El sistema verifica que los campos estén llenos , si no resulta , muestra un mensaje de error ” El campo USUARIO es obligatorio”	*Satisfactorio.*	
	V Yisell	V ycarias	N/A	I (no marcar ninguna opción)	N/A	V Femenino	El sistema verifica que los campos estén llenos , si no resulta , muestra un mensaje de error ” Debe escoger al menos un ROL”	*Satisfactorio*	

SC 2: Actualizar Datos del Usuario.

Escenario	Usuario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.1: Seleccionar el Usuario a modificar.	V	El sistema muestra los datos del usuario seleccionado en la lista desplegable.	*Satisfactorio.*	El Sistema muestra una lista desplegable donde se selecciona el usuario, muestra un formulario con todos los datos del usuario. El Sistema habilita la opción de Guardar y Enviar, muestra una ventana emergente solicitando confirmación de la operación, verifica que los datos estén correctos Almacena los Datos Actualizados y muestra un mensaje del éxito.
EC 1.2: No seleccionar el Usuario a modificar	I	El sistema valida que la información seleccionada sea correcta, en este caso no es así, entonces muestra un mensaje de error. "Debe seleccionar un usuario"	. No Satisfactorio	El sistema muestra el mensaje "Datos introducidos incorrectos" El Sistema muestra un mensaje de que no se pueden introducir letras/números/caracteres (&, %, etc.) en ese campo, limpia todos los caracteres extraños. El sistema no envía los datos y muestra un mensaje: "Los datos enviado son incorrectos". El Sistema colorea en color rojo los campos que contienen datos incorrectos.

SC 3: Listar/Eliminar Usuario.

Escenario	Usuario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 3.1: Listar / Eliminar el usuario satisfactoriamente.	V	El sistema lista todos los usuarios y elimina el seleccionado. . .	*Satisfactorio.*	El Sistema muestra una lista desplegable donde se selecciona el usuario, muestra un formulario con todos los datos del usuario. El Sistema habilita la opción de Guardar y Enviar, muestra una ventana emergente solicitando confirmación de la operación, verifica que los datos estén correctos Almacena los Datos Actualizados y muestra un mensaje del éxito de la operación.

Escenario	Usuario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 3.2: Listar / Eliminar el usuario dejando campos vacíos.	I	El sistema lista todos los usuarios y al no estar seleccionado ninguno a eliminar muestra mensaje de error" Debe escoger al menos un usuario"	No Satisfactorio	

Anexo 5: Caso de Prueba Gestionar Inventario Barrenas.

1. **Descripción de Variables.**

   SC: Insertar Inventario Barrena.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	No.	Campo de texto	No	Puede introducirse números y letras.
2	Tipo	Campo de texto	No	Puede introducirse letras y números.
3	No. Serie	Campo de texto	No	Permite insertar números y letras.
4	Fabricante	Lista desplegable	No	Lista desplegable con los posibles fabricantes.
5	IADC	Lista desplegable	No	Lista desplegable con las posibles numeraciones.
6	OD(mm)	Campo de texto	No	Campo de texto que permite solo numeraciones.
7	Longitud(m)	Campo de texto	No	Campo de texto que solo permite numeraciones.

SC 2: Modificar Datos en el Inventario Barrena				
1	No.	Campo de texto	No	Puede introducirse números y letras.
2	Tipo	Campo de texto	No	Puede introducirse letras y números.
3	No. Serie	Campo de texto	No	Permite insertar números y letras.
4	Fabricante	Lista desplegable	No	Lista desplegable con los posibles fabricantes.
5	IADC	Lista desplegable	No	Lista desplegable con las posibles numeraciones.
6	OD(mm)	Campo de texto	No	Campo de texto que permite solo numeraciones.
7	Longitud(m)	Campo de texto	No	Campo de texto que solo permite numeraciones.

1. **Matriz de Datos.**

SC 1: Insertar Datos en el Inventario Barrena.

Escenario	No	Tipo	No Serie	Fabricante	IADC	OD (mm)	Longitud (m)	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC: 1.1 Insertar los datos en el inventario satisfactoriamente.	V (2AR)	V (Smith)	V (PB-3481)	V (SMITH)	V (515)	V (445)	V (0.5)	El sistema comprueba los datos y los inserta en el inventario.	*Satisfactorio.*	1.1 El Sistema muestra un formulario solicitando los datos del inventario.2.1 El sistema verifica si los datos son correctos.3.1El Sistema muestra una venta emergente pidiendo confirmación de la operación. El Sistema comprueba que los datos no contengan errores. El Sistema almacena la información. 4.3 El Sistema envía un mensaje del éxito de la operación.
EC: 1.1 Insertar los datos en el inventario y que estos no sean correctos.	I (2AR&)	V (Smith)	V (PB-3481)	V (SMITH)	V (515)	V (445)	V (0.5)	El sistema comprueba que la información insertada sea correcta si no es así lanza un mensaje de error.	No Satisfactorio	1.1 El Sistema muestra un formulario solicitando los datos del inventario.2.1 El Sistema verifica si los datos son correctosa.1 “Datos introducidos incorrectos” El Sistema muestra un mensaje de que no se pueden introducir letras/números/caracteres (&, %, etc.) en ese campo. a.2 El Sistema limpia todos los caracteres
	V (2ARP)	I (Smith %)	V (PB-3485)	V (SMITH)	V (515)	V (445)	V (0.5)		No Satisfactorio	

Escenario	No	Tipo	No Serie	Fabricante	IADC	OD (mm)	Longitud (m)	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
	V (2ARL)	V (Smith)	I (PB-3481")	V (SMITH)	V (515)	V (445)	V (0.5)		No Satisfactorio	4b.1 El Sistema muestra un mensaje: "Los datos enviado son incorrectos" 4b.2 El Sistema colorea en color rojo los campos que contienen datos incorrectos.
	V (2ARM)	V (Smith)	V (PB-3481)	V (SMITH)	V (515)	I (no.)	I (no.)		No Satisfactorio	
EC: 1.2 Insertar los datos en el inventario dejando campos vacíos o nulos.	I (Vacío)	V (Smith)	V (PB-3481)	N/A	N/A	V (445)	V (0.5)	El sistema verifica que los campos estén llenos , si no resulta , muestra un mensaje de error "Se deben llenar todos los campos"	*Satisfactorio.*	.1 El Sistema muestra un formulario solicitando los datos del inventario.2.1 El Sistema verifica si los datos son correctos y los campos completados, en el caso de no ser así emite un mensaje de advertencia "Debe llenar los campos vacíos"
	V (2AR)	I (vacío)	V (PB-3481)	N/A	N/A	V (445)	V (0.5)		. *Satisfactorio.*	
	V (2AR)	V (Smith)	I (vacío)	N/A	N/A	V (445)	I (vacío)		. *Satisfactorio.*	
	V (2AR)	V (Smith)	V (PB-3481)	N/A	N/A	I (Vacío)	V (0.5)		*Satisfactorio.*	
	V (2AR)	V (Smith)	V (PB-3481)	N/A	N/A	V (445)	I (Vacío)		*Satisfactorio.*	

SC 2: Modificar Datos en el Inventario Barrena.

Escenario	No	Tipo	No Serie	Fabricante	IADC	OD(mm)	Longitud (m)	Respuesta del Sistema	Resultado de Prueba	Flujo Central
EC: 2.1 Modificar los datos correctamente.	V (3AR)	V (Hughes)	V (6681231)	V (REED)	V (117)	V (445)	V (0.5)	El sistema comprueba los datos modificados y los inserta en el inventario.	*Satisfactorio.*	El Sistema muestra un formulario con todas las barrenas que están inventariadas.2.1 El Sistema Muestra permite modificar la información de la barrena
EC: 2.2 Modificar los datos y que estos no se muestren o sean incorrectos.	I (3AR&)	V (Hughes)	V (6681231)	V (REED)	V (117)	V (445)	V (0.5)	El sistema comprueba que la información insertada sea correcta si no es así lanza un mensaje de error.	No Satisfactorio	1.1 El Sistema muestra un formulario solicitando los datos del inventario para modificar.2.1 El Sistema verifica si los datos son correctos, si no emite.a.1 “Datos introducidos incorrectos” El Sistema muestra un mensaje de que no se pueden introducir letras/números/caracteres (&, %, etc.) en ese campo. a.2 El Sistema limpia todos los caracteres extraños.4a.1 El Sistema no envía los datos 4b.1 El Sistema muestra un mensaje: “Los datos
	V (3AR)	I (Hughes %1)	V (6681231)	V (REED)	V (117)	V (445)	V (0.5)		. No Satisfactorio	
	V (3AR)	V (Hughes)	I (668123 1#)	V (REED)	V (117)	V (445)	V (0.5)		No Satisfactorio	
	V (3AR)	V (Hughes)	V (6681231)	V (REED)	V (117)	I (no.)	V (0.5)		. No Satisfactorio	
	V (3AR)	V (Hughes)	V (6681231)	V (REED)	V (117)	V (445)	I (no.)		No Satisfactorio	

Escenario	No	Tipo	No Serie	Fabricante	IADC	OD(mm)	Longitud (m)	Respuesta del Sistema	Resultado de Prueba	Flujo Central
EC: 2.3 Modificar los datos dejando sus campos vacíos o nulos.	I (Vacío)	V (Hughes)	V (6681231)	V (REED)	V (117)	V (445)	V (0.5)	El sistema verifica que los campos estén llenos , si no resulta , muestra un mensaje de error ”Se deben llenar todos los campos”	*Satisfactorio.*	.1 El Sistema muestra un formulario solicitando los datos del inventario.2.1 El Sistema verifica si los datos son correctos y los campos completados, en el caso de no ser así emite un mensaje de advertencia “Debe llenar los campos vacíos”
	V (3AR)	I (Vacío)	V (6681231)	V (REED)	V (117)	V (445)	V (0.5)		*Satisfactorio.*	
	V (3AR)	V (Hughes)	I (Vacío)	V (REED)	V (117)	V (445)	V (0.5)		*Satisfactorio.*	
	V (3AR)	V (Hughes)	V (6681231)	V (REED)	V (117)	I (Vacío)	V (0.5)		*Satisfactorio.*	
	V (3AR)	V (Hughes)	V (6681231)	V (REED)	V (117)	V (445)	I (Vacío)		*Satisfactorio.*	

Glosario de Términos.

Actividad: Unidad tangible de trabajo realizada por un trabajador en un flujo de trabajo, de forma que: (1) implica una responsabilidad bien definida para el trabajador, (2) produce un resultado bien definido (un conjunto de artefactos) basado en una entrada bien definida (otro conjunto de artefactos), (3) la ejecución de una operación, por un trabajador.

Artefacto: Pieza de información tangible que (1)es creada, modificada y usada por los trabajadores al realizar actividades;(2)representa un área de responsabilidad y (3)es candidata a ser tenida en cuenta para el control de la configuración. Un artefacto puede ser un modelo, un elemento de un modelo o un documento.

Caso de Prueba: Especificación de un caso para probar el sistema, incluyendo que probar, con que entradas y resultados y bajó que condiciones.

Cliente: Persona, organización o grupo de personas que encarga la construcción de un sistema, ya empezando desde cero, o mediante el refinamiento de versiones sucesivas.

Caso de Uso: Una descripción de un conjunto de secuencia de acciones, incluyendo variaciones, que un sistema lleva a cabo y que conduce a un resultado observable de interés para un actor determinado.

Defecto: Anomalía del sistema, por ejemplo un síntoma de error en el software descubierto durante las pruebas, o un problema descubierto durante una revisión.

Escenario: Una secuencia específica de acciones que ilustran un comportamiento.

Evaluación de las pruebas: Evaluación de los resultados, del esfuerzo de prueba, cómo cobertura de casos de pruebas y estado de defectos.

Flujo de trabajo: Cada uno de los flujos de trabajo, requisitos, análisis, diseño, implementación o pruebas. Puede describirse en términos de diagramas de actividades, que incluyen trabajadores participantes, las actividades que realizan y artefactos que producen.

Interfaz de Usuario: Interfaz a través del cual un usuario interactúa con el sistema.

Iteración: Conjunto de actividades llevadas a cabo de acuerdo a un plan y unos criterios de evaluación, que lleva producir una versión, ya sea interna o externa.

Lenguaje de Modelado Unificado (UML): Lenguaje estándar para el modelado de software, lenguaje para visualizar, especificar, construir y documentar los artefactos de un software.

Método: La implementación de una operación.

Plan de Pruebas: Plan que describe las estrategias, recursos y programación de las pruebas.

Pos condición: Una restricción que ha de ser cierta al completarse una operación.

Precondición: Una restricción que ha de ser cierta cuando una operación es invocada.

Proyecto: Esfuerzo de desarrollo para llevar un sistema a lo largo de un ciclo de vida.

Prueba (Flujo de trabajo): Flujo de trabajo fundamental, cuyo propósito esencial es orientar el desarrollo hacia el sistema correcto.

Requisito: Condición o capacidad que debe cumplir el sistema.

Requisito Funcional: Requisito que especifica una acción que debe ser capaz de realizar el sistema, especifica comportamiento de entrada/salida de un sistema.

Sistema: Una colección de subsistema organizados para llevar a cabo un propósito específico y descritos por un conjunto de modelos, posiblemente desde distintos puntos de vista.

Stakeholders: También llamados interesados o involucrados en un problema determinado que necesitan una solución óptima. Desde el punto de vista del desarrollo de sistemas, un Stakeholders es aquella persona o entidad que esta interesada en la realización de un proyecto o tarea, auspiciando el mismo ya sea mediante su poder de decisión o de financiamiento.

Subsistema: Una agrupación de elementos, de los que algunos elementos constituyen un especificación del comportamiento ofrecido por los otros elementos contenidos.

Trabajador: Puesto asignado a una persona o equipo, que tiene responsabilidad y habilidades para la realización de actividades.

Usuario: Humano que interactúa con un sistema.

Versión: Conjunto de artefactos relativamente completo y consistente.