UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 9

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

Título: Diseño y Aplicación de Casos de Prueba al Sistema de Gas Manufacturado

AUTORA: Diarilys Varela González.

TUTOR(a): Ing. Ismaila López Sotolongo.

Ing. Surima Ge Pérez.

CO-TUTORA: Ing. Yaquelin Cintra Almaguer.

Ciudad Habana, Junio de 2010

"Año 52 de la Revolución"

Dedicatoria

Le dedico mi tesis especialmente a mi abuela Laude por ser la persona que más amo en el mundo, por estar a mi lado en los buenos y malos momentos, dándome fuerza y ánimo para seguir adelante, sin ella se me hubiera hecho imposible llegar hasta aquí hoy. Gracias por estar siempre conmigo abuela.

A mi mamá por ser única para mí, por quererla tanto, por ser para mí la mejor madre del mundo, por entenderme y complacerme siempre en todo, por darme todo su amor y cariño, por hacer lo posible y lo imposible porque este día llegara. A ella y a mi abuela les debo cada día de estos 5 años de mi carrera.

A mi papá y a mi hermana por sus sabios consejos y por apoyarme en todo lo que necesito.

A Trini y a Carlos por brindarme su ayuda incondicional durante toda la carrera y por atenderme todos los fines de semana en su casa. Los voy a extrañar mucho.

A mis amigos Annies, Nuris, Yaimara, Yasmanys y César por apoyarme y estar conmigo siempre que los necesité, nunca los olvidaré.

En general la dedico a toda mi familia; a mis amigos por estar todos estos años a mi lado, por ayudarme y acompañarme en los buenos y malos momentos, a mis compañeros del aula con quienes he pasado estos 5 años de universidad; en fin a todos los que de una forma u otra han hecho posible que este día llegara.

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a La Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los 28 días del mes de Junio del año 2010.

_____	_____	_____
Firma del Autor	Firma del Tutor	Firma del Tutor
Diarilys Varela González	Ing. Ismaila López Sotolongo	Ing. Surima Ge Pérez

Resumen

La facultad 9 de la Universidad de las Ciencias Informáticas (UCI) desarrolla múltiples proyectos encaminados a la producción de software, entre ellos se encuentra el proyecto Sistema de Facturación y Cobro para la empresa de Gas Manufacturado. Este proyecto desarrolla el producto ManuGas.

El proyecto tiene como objetivo principal la implantación de un sistema informático que automatice los principales procesos de facturación y cobro de la empresa de Gas Manufacturado de Ciudad de La Habana. Una vez implementado el software es sometido a un proceso de prueba teniendo como objetivo principal detectar la mayor cantidad de defectos posibles. De ahí la entrega de un producto que cumpla con las especificidades y requerimientos solicitados por el cliente.

En el presente trabajo quedó detallado todo lo relacionado al proceso de prueba aplicado al producto ManuGas. Entre los principales aspectos tratados se encuentra una breve descripción de los niveles y métodos de pruebas existentes, así como la metodología de software seleccionada para garantizar la calidad y correcto funcionamiento de la aplicación, también se describió la estrategia de trabajo, se diseñaron los casos de pruebas correspondientes al modelo de caso de uso del sistema y se registraron los errores detectados en el informe de No Conformidades.

Finalmente se hizo un análisis sobre el proceso de prueba realizado con el propósito de evaluar la calidad del producto. Con la realización de las pruebas se pudo comprobar que las mismas arrojaron resultados puntuales, que sirvieron para mejorar la calidad del software y el correcto funcionamiento de la aplicación.

Palabras Claves

Aplicación, Calidad, Pruebas, Software.

Índice

Índice

Introducción

Se percibe en el mundo un incremento del uso de las tecnologías, que se refleja en la automatización de cada uno de los procesos que se ejecutan en las principales industrias de nuestra sociedad. Esto trae aparejado la existencia de nuevas empresas y compañías queriendo incursionar en la producción del software, por el mercado que en la actualidad se gana y representa la realización de productos informáticos.

La informatización de nuestros servicios no es un proceso arbitrario, sino que por el contrario incluye todo el proceso de supervisión y mejora del mismo, asegurando que se sigan los procedimientos acordados y evitando la no resolución de los problemas que puedan surgir, con el objetivo precisamente de llevar a cabo cada una de las acciones que garanticen la calidad y eficacia del software, lo que es muy importante a tener en cuenta en la realización de un producto pues es fundamental para su aceptación por parte del cliente.

No siempre se logra que el resultado final del producto cumpla los requisitos establecidos. Por tal motivo se le realizan pruebas al software durante el ciclo de desarrollo del mismo, lo que constituye un proceso aplicado al producto para corregir los errores que presenta.

Cuba es un país con disposición, conocimiento y posibilidades para avanzar en el mundo de la informática, por eso el mercado actual reclama software que de respuesta a sus necesidades, con requisitos cada vez más exigentes y con una elevada competitividad. Por tal motivo las empresas dedicadas a la producción de software tienen la necesidad de crear sus productos con la mayor eficiencia posible, aunque no todas demuestren esfuerzo y confianza en su desempeño.

Con la evolución de la informatización nace al calor de la batalla de ideas la universidad más joven del país: La Universidad de las Ciencias Informáticas (UCI). Institución que se dedica a la formación de profesionales Ingenieros en Ciencias Informáticas y a la producción de software.

Con este enfoque se creó en la facultad 9, en el Centro de Geoinformática y Señales Digitales (GEySED) el proyecto Sistema de Facturación y Cobro para la Empresa de Gas Manufacturado; dentro de sus principales objetivos está la implantación de un sistema informático que automatice los principales procesos de la empresa de Gas Manufacturado de Ciudad de La Habana permitiendo:

- Agilizar el envío de información de las Casas Comerciales a la Unidad Empresarial de Base (UEB).
- Efectuar el cobro de un cliente en cualquier Casa Comercial.
- Consultar en tiempo real la información actual de los procesos de Facturación y Cobro en la empresa.
- Efectuar modificaciones de cualquier tipo a sus clientes.
- Emitir los recibos de cobro del gas manufacturado.
- Visualizar e imprimir los reportes del proceso de facturación y cobro.
- Admitir la realización de cuadres parciales, que permitan saber en cada momento el estado de gestión de la actividad comercial.
- Mantener el control de las tareas que se realizan sobre la información, a través del acceso de usuarios al sistema.

Al hacer un estudio de lo anteriormente expuesto, se pudo comprobar que existe la necesidad de facturar el servicio de gas a los clientes por una tarifa escalonada, puesto que el uso de la tarifa fija implantada anteriormente, imposibilita a los empleados de la empresa efectuar el cobro y el pago respectivamente del servicio del gas manufacturado a partir de lo que realmente consumen y no por una tarifa dinámica. Utilizan para ello un Sistema de Metrado (SISMET), desarrollado en Visual Basic 6, que presenta algunas dificultades en las actividades de cierre con el sistema, provocando atrasos en las operaciones de la empresa. Además no tienen un mecanismo implementado que permita facilitar la gestión de cobro a los clientes, entre otras dificultades menores.

El impacto provocado por el sistema anteriormente planteado, no les permite a los clientes pagar en otra Casa Comercial que no sea la que le corresponde. Ocasiona altos tiempos de respuesta y dificultades para consolidar la información proveniente de las Casas Comerciales. No permite cobrar por una tarifa escalonada. Imposibilita consultar toda la información referente a los procesos de Cobro y Facturación por los directivos de la UEB. Evita conocer el estado de las Casas Comerciales en tiempo real y no permite saber el porciento de clientes facturados totales ni por cada Casa Comercial.

Introducción

Para que el producto llegue con la calidad requerida a manos del usuario, se hace imprescindible realizar un proceso de prueba que tenga como objetivo principal detectar la mayor cantidad de defectos posibles en el software. De ahí la entrega de un producto que cumpla con las especificidades y requerimientos solicitados por el cliente.

Teniendo en cuenta los aportes del proyecto a la Empresa de Gas Manufacturado y las actividades que hasta ahora se han desarrollado en el mismo en aras de garantizar la calidad y correcto funcionamiento de cada una las funcionalidades que han sido implementadas, se hace necesario llevar a cabo una de las acciones, que puede garantizar la detección y posterior corrección de los defectos que se evidencien en la aplicación, constituyendo de esta forma, para la presente investigación, el **problema a resolver:** ¿Cómo detectar los defectos en la aplicación de Gas Manufacturado?

Para llevar a cabo la verificación y validación del producto se debe realizar un estudio de todas las pruebas existentes permitiendo conocer los errores sobre los cuales trabajar, obteniendo así un producto más eficiente, por lo que el **objeto de estudio** de la investigación es: El proceso de prueba como parte del desarrollo del software; enmarcando así el **campo de acción en**: La aplicación de casos de pruebas al producto Gas Manufacturado. Teniendo en cuenta lo anteriormente expuesto el **objetivo general** es: Diseñar y aplicar los casos de prueba al producto Gas Manufacturado.

La **hipótesis** planteada es: Si se diseñan y aplican casos de prueba bien definidos al producto ManuGas, entonces se podrán detectar los defectos de la aplicación, contribuyendo a la calidad de la misma.

Para dar cumplimiento al objetivo general se trazaron las siguientes **tareas de investigación:**

- Caracterizar el proceso de prueba según la metodología RUP.
- Describir los métodos de pruebas de software.
- Diseñar y aplicar casos de pruebas.
- Elaborar el plan de prueba.
- Evaluar el resultado de las pruebas.

Una vez concluida la investigación se espera obtener como **resultado:**

- Diseño de casos de pruebas a la aplicación ManuGas.
- Plan de pruebas del producto ManuGas.
- Evaluación de las pruebas realizadas.

Para darle cumplimiento a los objetivos trazados en la investigación, se utilizaron los siguientes métodos científicos:

Métodos teóricos:

Histórico-lógico: Para investigar la información que se tiene hasta el momento sobre el estado del arte de las pruebas de software, las características fundamentales que ponen de manifiesto su desarrollo y hacer un resumen de los aspectos más importantes durante el proceso de la investigación.

Analítico-sintético: Durante la primera parte de la investigación se utilizó este método para hacer un análisis bibliográfico del tema relacionado con las pruebas de software, con el objetivo de realizar una síntesis de los aspectos necesarios para la llevar a cabo el diseño y aplicación de las pruebas realizadas al producto ManuGas.

Método de modelación: Durante el diseño de los casos de prueba a aplicarse al producto ManuGas se utilizó el método de modelación, con el objetivo de facilitar la posterior implementación de éstos, partiendo de la descripción de cada uno de los casos de uso del sistema del proyecto.

Métodos empíricos:

Observación: Durante la investigación se utilizó este método para conocer la realidad del proceso de prueba, teniendo como base el ambiente de trabajo del proyecto y sus procesos fundamentales.

Entrevista: Se utilizó de forma no estructurada el método de la entrevista para obtener información clara y precisa sobre la descripción de los casos de usos del sistema, así como de los conceptos y aspectos de mayor envergadura que se utilizan en el ambiente de trabajo del proyecto.

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

Capítulo 1: En este capítulo se desarrolla la fundamentación teórica, donde se explican los conceptos asociados al proceso de pruebas de software. Se caracteriza el proceso de prueba y los roles de diseñador de caso de prueba y probador. Se hace énfasis en todos los aspectos importantes para el diseño y aplicación de las pruebas y se hará referencia a las definiciones que propone RUP como metodología de desarrollo para este proceso.

Capítulo 2: En este capítulo se realiza la planificación y el diseño de las pruebas al software. Se hace una descripción del producto. Se presenta el plan del sistema definido por el proyecto y el diseño de los casos de pruebas realizados a la aplicación informática ManuGas. Se hará una descripción de la estrategia de las pruebas realizadas, así como los niveles y métodos seleccionados.

Capítulo 3: Este capítulo se denomina: Ejecución y Evaluación de las pruebas al software, muestra los resultados alcanzados durante la ejecución de las pruebas, así como una evaluación de los mismos, documentándose para su posterior corrección en la aplicación por el equipo de desarrollo del proyecto.

Capítulo 1: Fundamentación Teórica

Introducción

El proceso de desarrollo de un software consta de una serie de etapas, a medida que se desarrolla cada una, se va conformando el producto final. Los procesos de prueba son de gran importancia dentro del ciclo de vida de desarrollo del software, son herramientas que aseguran que un producto cumpla los requisitos esperados. Las pruebas logran depurar el producto desde sus inicios para que llegue a manos de los usuarios finales cumpliendo con todas sus funcionalidades y con la menor cantidad de defectos posibles.

En el presente capítulo se abordarán conceptos y definiciones relacionados con la calidad, así como los factores que miden la misma. Se realizará un estudio sobre las pruebas en el desarrollo del software, las características de las pruebas, los niveles, los métodos, el enfoque de las pruebas, el plan de prueba y se hará referencia a los documentos del proceso de pruebas utilizados en la universidad. También se hará énfasis en el diseño de prueba y en la estrategia de pruebas de software. Además se caracterizarán los roles diseñador de prueba y probador. Por último se hará referencia a las definiciones que propone RUP como metodología de desarrollo para este proceso.

1.1 Conceptos Fundamentales

- **Calidad**

"Conjunto de propiedades y de características de un producto o servicio que le confieren aptitud para satisfacer unas necesidades explícitas o implícitas". (1)

La calidad es el conjunto de cualidades de un software que determinan su utilidad y existencia. Es el procedimiento enfocado al perfeccionamiento del producto o servicio para satisfacer las demandas y necesidades de los clientes.

- **Calidad del software**

La calidad del software es un aspecto muy particular que debe tener cada producto software basándose en la eficiencia del mismo, con el objetivo de satisfacer las necesidades y expectativas del cliente.

Pressman define calidad de software como la “Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente”. (2)

- **Aseguramiento de la calidad del software**

“El aseguramiento de calidad del software es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto (software) satisfaga los requisitos dados de calidad”. (3)

El aseguramiento de la calidad es una forma o acción de llevar a cabo un producto con el fin de demostrar el respaldo necesario del mismo y transmitir seguridad sobre la eficiencia del software a los usuarios finales.

- **Pruebas**

Una prueba es un proceso para ejecutar un programa con el fin de encontrar errores. Es una actividad de evaluación del producto, realizada para saber en qué condiciones se encuentra el software. Permiten al desarrollador determinar si el producto generado satisface las expectativas establecidas por el cliente.

“Una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto”. (4)

- **Caso de uso**

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importatante. Los casos de uso representan los requisitos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso. (5)

Un caso de uso no es más que la interacción que se establece entre un sistema y sus actores. Indican como deberia actuar el sistema con el usuario o con otro sistema para conseguir un objetivo en específico. Además se centra en alcanzar una meta o tarea del negocio.

- **Caso de prueba**

Los casos de prueba son un conjunto de condiciones que determinarán si los requisitos establecidos por los usuarios finales de una aplicación son completamente satisfactorios, con el objetivo de detectar la mayor cantidad de defectos posibles en el software y satisfacer las necesidades del cliente.

Un 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 que probarse. (5)

- **Plan de prueba**

El plan de prueba describe las estrategias, recursos y planificación de la prueba. La estrategia de prueba incluye la definición del tipo de prueba a realizar para cada iteración y sus objetivos, el nivel de cobertura de prueba y de código necesario y el porcentaje de pruebas que deberian ejecutarse con un resultado específico. (5)

Un plan de prueba es un documento que puede desarrollarse durante el proceso de desarrollo de las pruebas. Proporciona el marco dentro del cual, el equipo de prueba desarrolla las pruebas trabajando con los recursos y la planificación adecuada.

- **Evaluación de la prueba**

Una evaluación de prueba es una evaluación de los resultados de los esfuerzos de prueba, tales como la cobetura del caso de prueba, del código y el estado de los defectos. (5)

1.2 Factores que miden la calidad del software

Un aspecto importante a tener en cuenta en el desarrollo de un producto de software son los factores que miden su calidad. Estos se centran en 3 categorías:

❖ **Según sus características operativas:** (6)

- **Corrección:** hasta donde satisface un programa su especificación y logra los objetivos del cliente.
- **Fiabilidad:** hasta dónde se puede esperar que un programa lleve a cabo la función con la que fue concebido, con la exactitud requerida.
- **Eficiencia:** la cantidad de recursos informáticos y de código necesarios para que un programa funcione.
- **Seguridad:** hasta dónde se puede controlar el acceso al software o a los datos por personas no autorizadas.
- **Facilidad de uso:** aprender a operar los datos de entrada e interpretar las salidas de un programa.

❖ **Según la capacidad de soportar los cambios:** (6)

- **Facilidad de mantenimiento:** la capacidad para localizar y arreglar un error en un programa.
- **Flexibilidad:** el esfuerzo necesario para modificar un programa operativo.
- **Facilidad de prueba:** el esfuerzo necesario para probar un programa para asegurarse de que realiza su función pretendida.

❖ **Según la adaptabilidad a nuevos entornos:** (6)

- **Portabilidad:** la facilidad para transferir el programa de un entorno de sistema hardware y/o software a otro entorno diferente.
- **Reusabilidad (Capacidad de reutilización):** hasta dónde se puede volver a emplear un programa (o partes de un programa) en otras aplicaciones.
- **Interoperabilidad:** el esfuerzo para acoplar un sistema con otro.

1.3 Las pruebas en el desarrollo del software

Cuando se habla de calidad es importante mencionar las pruebas de software, éstas juegan un papel fundamental durante el ciclo de desarrollo del mismo. La etapa de prueba puede llegar a ser la más lenta del proceso debido a la complejidad de probar una aplicación, sin embargo si el proceso se desarrolla siguiendo una planificación y la metodología adecuada, se pueden lograr importantes disminuciones en los costos de desarrollo y en la cantidad de errores.

El objetivo de las pruebas es asegurar que el software cumpla con las especificaciones requeridas y eliminar los posibles defectos que este pudiera tener, la experiencia parece indicar que donde hay un defecto hay otros, es decir, la probabilidad de descubrir nuevos defectos en una parte del software es proporcional al número de defectos ya descubierto.

Un proceso de prueba será exitoso cuando encuentre errores, los que no son siempre fruto de la negligencia del programador.

De forma general las pruebas deben centrarse en los siguientes objetivos: (5)

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias solo al final de la iteración.
- Diseñar e implementar las pruebas creando los casos de prueba que especifiquen qué probar, creando los procedimientos de prueba que especifiquen cómo realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas.
- Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados.

Las pruebas son elementos críticos para determinar la calidad del software. La importancia de los costos asociados a los errores promueve la definición y aplicación de un proceso de prueba minuciosa y bien planificada. Las pruebas a pesar de que no pueden asegurar la ausencia de defectos, sino demostrar que existen defectos en el software, permiten validar y verificar el producto, entendiendo como validación el proceso que determina si el software satisface los requisitos, mientras que la verificación es el proceso que determina si los productos de una fase satisfacen las condiciones de dicha fase.

1.4 Características de las pruebas

Las pruebas se deben realizar, en el entorno en el que se utilizará el sistema, lo que incluye el personal que lo maneja. Las pruebas no comienzan formalmente hasta que un número mínimo predeterminado ha instalado el software. En particular, usuarios que tardan en comenzar, generalmente nunca lo hacen y ponen en peligro todo el proceso.

Las características de las pruebas son las siguientes:

- Una buena prueba tiene una alta probabilidad de encontrar un error. El ingeniero de software debe tener un alto nivel de entendimiento del software a construir para poder diseñar buenos casos de prueba que encuentren el mayor número de defectos. (7)
- Una buena prueba no debe ser redundante. Uno de los objetivos de las pruebas es encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles, por lo cual no se deben diseñar casos de prueba que tengan el mismo propósito que otros sino que se debe buscar diseñar el menor número de casos de prueba que permitan probar adecuadamente el software y que permitan optimizar los recursos. (7)
- Una buena prueba debería ser la mejor de la cosecha. La limitación en tiempo y recursos puede impedir que se ejecuten todos los casos de prueba de un grupo de pruebas similares por lo cual en estos casos se debería seleccionar la prueba que tenga la mayor probabilidad de descubrir errores. (7)
- Una buena prueba no debería ser ni demasiado sencilla ni demasiado compleja. (7)

1.5 Niveles de prueba

El producto ManuGas debe de cumplir una serie de requisitos. Para poder evaluar a este producto se hacen varias pruebas en dependencia del nivel en que se encuentre el proceso de prueba.

Los tipos de pruebas son:

- **Prueba Unitaria:**

Es la escala más pequeña de la prueba, está basada en la funcionalidad de los módulos del programa. Su objetivo es probar el comportamiento de cada uno de los componentes de forma independiente, por lo que debe realizarse una vez sea implementado el componente y se prueba la funcionalidad de una clase o conjunto de clases que se correlacionan. (8)

Para probar los componentes implementados como unidades individuales, se realizan las pruebas de especificación o de caja negra, y pruebas de estructuras, o de caja blanca. (8)

- **Prueba de Integración:**

Las pruebas de integración son técnicas sistemáticas para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. Su objetivo fundamental es probar la unión de los componentes del sistema, una vez estos hayan rebasado las pruebas unitarias, se deberá verificar que estos interactúan correctamente a través de las funcionalidades expuestas en sus interfaces, este tipo de prueba deberá realizarse durante la fase de construcción, inmediatamente se haya implementado el componente. (8)

Los tipos de pruebas de integración son los siguientes:

- Prueba Integración Descendente.

  La Prueba de Integración Descendente es un planeamiento incremental a la construcción de la estructura de programas. Se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el módulo de control principal (programa principal). (8)

- Prueba Integración Ascendente.

La Prueba Integración Ascendente, como su nombre lo indica, empieza la construcción y la prueba con los módulos atómicos (es decir, módulos de los niveles más bajos de la estructura del programa). (8)

- Prueba de Regresión.

La Prueba de Regresión es volver a ejecutar un subconjunto de pruebas que se han llevado a cabo anteriormente para asegurarse de que los cambios no han propagado efectos colaterales no deseados. (8)

- Prueba de Humo.

La Prueba de Humo es un método de Prueba de Integración que es comúnmente utilizada cuando se ha desarrollado un producto software empaquetado. Es diseñado como un mecanismo para proyectos críticos por tiempo, permitiendo que el equipo de software valore su proyecto sobre una base sólida. (8)

❖ **Prueba del Sistema**

La Prueba del Sistema, está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadoras. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas. (8)

Entre las pruebas del sistema se pueden considerar las siguientes:

- Prueba de Recuperación.

La Prueba de Recuperación es una prueba que se le hace al sistema que fuerza el fallo del software de muchas formas y verifica que la recuperación se lleve a cabo apropiadamente. (8)

- Prueba de Seguridad.

La Prueba de Seguridad intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán de accesos impropios. (8)

- Prueba de Resistencia.

  La Prueba de Resistencia ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales. (8)

- Prueba de Rendimiento.

  La Prueba de Rendimiento está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. (8)

- Prueba de instalación

  Se centra en asegurar que el sistema software desarrollado se puede instalar en diferentes configuraciones hardware y software y bajo condiciones excepciones, por ejemplo con espacio de disco insuficiente o continuas interrupciones. (8)

- Prueba de funcionalidad

  La prueba de funcionalidad fija su atención en la validación de las funciones, métodos, servicios, caso de uso. (8)

Las pruebas del sistema se usan para probar que el sistema funciona correctamente como un todo. Cada prueba de sistema prueba principalmente combinaciones de casos de usos instanciados bajo condiciones diferentes. Estas condiciones incluyen diferentes configuraciones hardware (procesadores, memoria principal, discos duros, etc.), diferentes niveles de carga del sistema, diferentes números de actores y diferentes tamaños de la base de datos. (8)

Entre los tipos de prueba que se realizan en un sistema, está el tipo que evalúa la funcionalidad de éste. La prueba de funcionalidad será aplicada al producto ManuGas, teniendo en cuenta que su funcionamiento se basa principalmente en la validación de los casos de usos.

El objetivo de las pruebas funcionales es asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados. Su meta es verificar el procesamiento, recuperación e implementación adecuada de las reglas del negocio y verificar la apropiada aceptación de datos. Su enfoque se basa en los requisitos funcionales (Casos de Uso) y las reglas del negocio.

La técnica usada por este tipo de prueba es el método de caja negra. Se ejecuta cada caso de uso, flujo de caso de uso o función, usando datos válidos e inválidos para verificar que se aplique apropiadamente cada regla del negocio. Que los resultados esperados ocurran cuando se usen datos válidos y que sean desplegados los mensajes apropiados de error y precaución cuando se usan datos inválidos. (9)

- **Pruebas de Validación**

La Prueba de Validación, es tan importante como el resto de las pruebas, y están presentes en todas las fases del desarrollo del sistema, su objetivo principal, como su nombre lo indica, no es más que validar, certificar y autenticar todas las funcionalidades del sistema. De ellas se obtienen información útil, que servirá para validar la implementación de los algoritmos. (8)

Los tipo de pruebas de validación son:

- Pruebas Alfa y Beta.

  Es un proceso que llevan a cabo los desarrolladores para descubrir errores que parezca que solo el usuario final puede descubrir. La Prueba Alfa se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso. La prueba Alfa se lleva a cabo en un entorno controlado. La Prueba Beta es realizada por los usuarios finales del software en los lugares de trabajos de los clientes. A diferencia de la Prueba Alfa, el desarrollador no está presente normalmente. Así, la Prueba Beta es una aplicación en vivo del software en un entorno que no puede ser controlado por el desarrollador. (8)

1.6 Documentos del proceso de pruebas utilizados en la UCI

En el proceso de prueba a realizar al producto ManuGas quedan definidos los mismos entregables que define el Centro para la Excelencia en el Desarrollo de Proyectos Tecnológicos de la UCI (CaliSoft):

1.6.1 El plan de prueba

El plan de prueba forma la infraestructura dentro de la cual el equipo que realiza las pruebas trabajará durante la planificación determinada. Dirige, orienta y restringe el esfuerzo de prueba, centrándose el trabajo en los entregables útiles y necesarios. Es un artefacto que define los objetivos de calidad, métodos y cronograma de las pruebas en el ámbito de la iteración (o el proyecto), los elementos de destino y el enfoque que se adopta. (11)

1.6.2 Caso de prueba

Los casos de prueba listan los ítems específicos que serán probados y describe los pasos detallados que serán seguidos para verificar el software. Son un conjunto de condiciones o variables bajo las cuales el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio, con el propósito de comprobar que todos los requisitos de la aplicación sean revisados.

1.6.3 El Informe de No Conformidades

El informe de no conformidades es un documento mediante el cual se van a archivar los incumplimientos de los requisitos que deben ser cumplidos en el proyecto. Este documento define el procedimiento interno a seguir por los desarrolladores del proyecto para el tratamiento de los errores cometidos. Es el resultado de un proceso que no cumple con los requisitos establecidos por el cliente.

1.7 Diseño de pruebas

Los métodos y estrategias de diseño de casos de prueba tienen como objetivo conseguir una confianza aceptable en que se detectarán los defectos existentes (ya que la seguridad total sólo puede obtenerse de la prueba exhaustiva, que no es practicable) sin que se necesite consumir una cantidad excesiva de recursos.

1.7.1 Enfoque de pruebas

Para el diseño de casos de prueba, se identifican 3 enfoques de pruebas: (7)

- **El enfoque funcional o de caja blanca:** consiste en centrarse en la estructura interna (implementación) del programa para elegir los casos de prueba.

- **El enfoque estructural o de caja negra:** consiste en estudiar la especificación de las funciones, la entrada y la salida para derivar los casos.

  - ✓ Este tipo de enfoque será el utilizado para la realización de los casos de pruebas**,** teniendo en cuenta su funcionamiento en la aplicación.

- **El enfoque aleatorio:** consistente en utilizar modelos (en muchas ocasiones estadísticos) que representen las posibles entradas al programa para crear a partir de ellos los casos de prueba.

1.8 Métodos de pruebas

Un método es una serie de pasos sucesivos que conducen a una meta. Existen dos tipos de métodos para probar un software.

1.8.1 Método de prueba de Caja Blanca

La prueba de caja blanca denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. (10)
Mediante este método el ingeniero de software puede obtener casos de prueba que:

- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.

- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.

- Ejecuten todos los bucles en sus límites y con sus límites operacionales.

- Ejerciten las estructuras internas de datos para asegurar su validez.

1.8.2 Método de prueba de Caja Negra

Las pruebas de caja negra se centran en los requisitos funcionales del software. Es decir, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Se trata de un enfoque que intenta descubrir diferentes tipos de errores que no se encuentran con los métodos de caja blanca. (10)

Las pruebas de Caja Negra, intenta encontrar errores de las siguientes categorías:

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

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

- **Métodos de prueba basados en grafos:** en este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. (10)

- **Partición equivalente:** método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. (10)

❖ **Análisis de valores límite:** los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites (AVL) como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límite. (10)

❖ **Prueba de la tabla ortogonal:** hay aplicaciones donde el número de parámetros de entrada es pequeño y los valores de cada uno de los parámetros está claramente delimitado. Cuando estos números son muy pequeños (por ejemplo, 3 parámetros de entrada tomando 3 valores diferentes), es posible considerar cada permutación de entrada y comprobar exhaustivamente el proceso del dominio de entrada. En cualquier caso, cuando el número de valores de entrada crece y el número de valores diferentes para cada elemento de dato se incrementa, la prueba exhaustiva se hace impracticable. (10)

❖ **Adivinando el error:** dado un programa particular, se conjetura, por la intuición y la experiencia, ciertos tipos probables de errores y entonces se escriben casos de prueba para exponer esos errores. Es difícil dar un procedimiento para esta técnica puesto que es en gran parte un proceso. (10)

1.9 Estrategia de prueba de software

Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la construcción correcta del software.

Las características generales son:

- ❖ La prueba comienza en el nivel de módulo y trabaja “hacia afuera”.
- ❖ En diferentes puntos son adecuadas a la vez distintas técnicas de prueba.
- ❖ La prueba y la depuración son actividades diferentes.
- ❖ Para conseguir estos objetivos el flujo de trabajo de Prueba 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.

1.9.1 Estrategia de aplicación de las pruebas

Algunas etapas típicas más en detalles del desarrollo de las pruebas de software son las siguientes: (4)

- ❖ Con el esquema del diseño del software, los módulos probados se integran para comprobar sus interfaces en el trabajo conjunto (prueba de integración).
- ❖ El software totalmente ensamblado se prueba como un conjunto para comprobar si cumple o no tanto los requisitos funcionales como los requisitos de rendimientos, seguridad, etc. (prueba funcional o de validación).
- ❖ El software ya validado se integra con el resto del sistema (por ejemplo, elementos mecánicos, interfaces electrónicas, etc.) para probar su funcionamiento conjunto (prueba del sistema).
- ❖ Por último, el producto final se pasa a la prueba de aceptación para que el usuario compruebe en su propio entorno de explotación si lo acepta como está o no (prueba de aceptación).

Teniendo en cuenta la estrategia anteriormente expuesta, se identifican los siguientes aspectos para el diseño de caso de prueba del producto ManuGas:

- ❖ Especificar los requisitos del producto de manera cuantificable mucho antes de que comiencen las pruebas.
- ❖ Establecer los objetivos de la prueba de manera explícita.
- ❖ Construir un software robusto, diseñado para probarse a sí mismo.

- Comprender qué usuarios van a manejar el software y desarrollar un perfil para cada categoría de usuario.

- Llevar a cabo revisiones técnicas formales, para evaluar la estrategia de prueba y los propios casos de prueba.

1.10 Metodologías de desarrollo de software

En la actualidad existen diferentes metodologías de desarrollo de software. Constituyen un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Se clasifican en metodologías ágiles o robustas. SCRUM y XP son conocidas como metodologías ágiles. Estas intentan evitar los caminos pesados de las metodologías tradicionales, enfocándose en las personas y en los resultados.

1.10.1 SCRUM

Esta metodología se basa en una filosofía del desarrollo ágil, creado por Hirotaka Takeuchi e Ikujiro Nonaka por el año 1986, para desarrollo de software. SCRUM aunque puede ser usado para otro tipo de proyectos y tiene demostrada efectividad en otras áreas, aunque generalmente es funcional solo para desarrollos de software porque para eso fue diseñado. La idea es desarrollar aplicaciones mucho más rápido y eficazmente. Esta metodología se basa principalmente en que los individuos están por encima de los procesos y las herramientas, en entregar soluciones por encima de reportes de seguimiento y en dar respuesta a los cambios, en lugar de ceñirse a seguir un plan. (12)

1.10.2 Programación Extrema (XP)

La Programación Extrema o Extreme Programming (XP) es un enfoque de la ingeniería de software formulado por Kent Beck y De Jean. Es la más destacada de los procesos ágiles de desarrollo de software. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software. (13)

1.10.3 Proceso Unificado de Modelado (RUP)

Sus siglas en inglés significan Rational Unified Process (Proceso Unificado de Modelado), es un producto del proceso de ingeniería de software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de alta calidad que resuelve las necesidades de los usuarios dentro de un presupuesto y tiempo establecido.

RUP es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. No es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. (13)

Características esenciales que definen al RUP

Los autores de RUP destacan que el proceso de software tiene 3 características esenciales:

Proceso Dirigido por los Casos de Uso: Con esto se refiere a la utilización de los Casos de Uso para el desenvolvimiento y desarrollo de las disciplinas con los artefactos, roles y actividades necesarias. Los Casos de Uso son la base para la implementación de las fases y disciplinas del RUP. Un Caso de Uso es una secuencia de pasos a seguir para la realización de un fin o propósito, y se relaciona directamente con los requerimientos, ya que un Caso de Uso es la secuencia de pasos que conlleva la realización e implementación de un Requerimiento planteado por el Cliente. (14)

Proceso Iterativo e Incremental: Es el modelo utilizado por RUP para el desarrollo de un proyecto de software. Este modelo plantea la implementación del proyecto a realizar en Iteraciones, con lo cual se pueden definir objetivos por cumplir en cada iteración y así poder ir completando todo el proyecto iteración por iteración, con lo cual se tienen varias ventajas, entre ellas se puede mencionar la de tener pequeños avances del proyectos que son entregables al cliente el cual puede probar mientras se está desarrollando otra iteración del proyecto, con lo cual el proyecto va creciendo hasta completarlo en su totalidad. (14)

Proceso Centrado en la Arquitectura: Define la Arquitectura de un sistema, y una arquitectura ejecutable construida como un prototipo evolutivo. Arquitectura de un sistema es la organización o estructura de sus partes más relevantes. Una arquitectura ejecutable es una implementación parcial del sistema, construida para demostrar algunas funciones y propiedades. RUP establece refinamientos sucesivos de una arquitectura ejecutable, construida como un prototipo evolutivo. (14)

1.11 Proceso de pruebas de RUP

Durante la fase de Inicio puede hacerse parte de la planificación inicial de las pruebas cuando se define el ámbito del sistema. Sin embargo, las pruebas se llevan a cabo sobre todo cuando un producto de desarrollo software es sometido a pruebas de integración y de sistema. Esto quiere decir que la realización de pruebas se centra en la fase de Elaboración cuando se prueba la línea base ejecutable de la arquitectura, mientras que en la fase de construcción se prueba cuando el grueso del sistema está implementado. Durante la fase de Transición el centro de las pruebas se desplaza hacia la corrección de defectos y a las pruebas de regresión, durante los primeros usos del sistema. (5)

Debido a la naturaleza iterativa del esfuerzo de desarrollo, algunos de los casos de prueba que especifican cómo probar los primeros productos de desarrollo software pueden ser utilizadas también como casos de prueba de regresión que especifican cómo llevar a cabo las pruebas de regresión sobre los productos de desarrollo software siguientes. El número de pruebas de regresión necesarias crece por tanto de forma estable a lo largo de las iteraciones, lo que significa que las últimas iteraciones requerirán un gran esfuerzo en pruebas de regresión. Es natural mantener el modelo de pruebas a lo largo del ciclo de vida del software completo, aunque el modelo de pruebas cambia constantemente debido a:

- La eliminación de casos de prueba obsoletos.
- El refinamiento de algunos casos de prueba en casos de prueba de regresión.
- La creación de nuevos casos de prueba para cada nuevo producto de desarrollo de software. (5)

RUP define cómo ejecutar el proceso de prueba teniendo en cuenta un grupo de artefactos, actividades y trabajadores. El Proceso Unificado de Desarrollo de Software plantea 7 artefactos para el flujo de trabajo de prueba, específicamente en este trabajo se generan los artefactos plan de prueba, casos de pruebas y evaluación de las pruebas.

1.11.1 Trabajadores del flujo de trabajo de prueba

RUP establece los siguientes trabajadores para el flujo de trabajo de prueba:

- Diseñador de pruebas.
- Ingeniero de componentes.
- Ingeniero de pruebas de integración.
- Ingeniero de pruebas de sistema.

RUP establece las fases de Inicio, Elaboración, Construcción y Transición para cada flujo de trabajo. El proyecto Sistema de Facturación y Cobro para la empresa de Gas Manufacturado establece las fases de Construcción y Transición para el flujo de trabajo de prueba, entre los que se encuentran los siguientes trabajadores:

- Administrador de Prueba.
- Analista de Prueba.
- Diseñador de Prueba.
- Probador.

Para el desarrollo del trabajo de diploma se realizó un estudio de los trabajadores que propone RUP y los que están establecidos en el proyecto y se llegó a la conclusión de que los necesarios para diseñar y aplicar las pruebas al producto ManuGas son diseñador de prueba y probador, ambos roles van a ser desarrollados por la misma persona.

1.11.2 Diseñador de Prueba

Un diseñador de prueba es responsable de la integridad del modelo de prueba, asegurando que el modelo cumple con su propósito. Los diseñadores de pruebas también planean las pruebas, lo que significa que deciden los objetivos de pruebas apropiados y la planificación de las pruebas. Además, los diseñadores de pruebas seleccionan y describen los casos de prueba y los procedimientos de prueba correspondientes que se necesitan, y son responsables de la evaluación de las pruebas de integración y de sistema, cuando éstas se ejecutan. (5)

1.11.3 Probador

Este rol es el responsable de las principales actividades durante la ejecución de las pruebas. Incluye la conducción de las pruebas necesarias y el registro de resultados de las mismas. Ejecuta las pruebas diseñadas y anota los resultados obtenidos.

Conclusiones Parciales

El análisis realizado en este capítulo ha mostrado claramente la importancia de la calidad de los productos en la industria del software. Para que un producto logre satisfacer a los clientes es fundamental un diseño de caso de prueba efectivo que permita detectar la mayor cantidad de errores en el software.

Se ha detallado los conceptos y definiciones relacionados con la calidad, así como los factores que miden la misma. Se realizó un estudio sobre las pruebas en el desarrollo del software, las características de las pruebas, los niveles, los métodos, el enfoque de las pruebas, el plan de prueba y se hizo referencia a los documentos del proceso de pruebas utilizados en la universidad. También se hizo énfasis en el diseño de prueba y en la estrategia de pruebas de software. Se caracterizó el rol diseñador de caso de prueba y probador y se hizo referencia a las definiciones que propone RUP como metodología de desarrollo para este proceso.

Teniendo en cuenta los resultados de la investigación y estando definido ya los niveles, los métodos y herramientas a utilizar, se da paso al diseño y aplicación de las pruebas de software.

Capítulo 2: Planificación y Diseño de las Pruebas al Software

Introducción

La fase de prueba es muy importante, puesto que en ella se refleja la calidad con que ha sido llevada a cabo la proyección del sistema. Cada proceso de prueba se inicia con la elaboración del plan de prueba, en el cual queda registrado el propósito de las pruebas así como los recursos y pasos a seguir para su desarrollo.

Anteriormente se analizaron los conceptos fundamentales relacionados con las pruebas de software, así como aspectos de gran importancia para su desarrollo. Lo planteado constituye una guía para lo expuesto en este capítulo. Se hará una breve descripción del sistema a probar y de las características del mismo. Se hará énfasis en los objetivos, alcance y estructura del plan de prueba del producto ManuGas. Se realizará una descripción de la estrategia de prueba realizada para el diseño y aplicación de las mismas, así como se identificará los niveles y métodos de pruebas seleccionados. Por último se hará referencia a la estructura de la planilla del diseño de caso de prueba realizado al producto ManuGas.

2.1 Descripción del sistema a probar

El proyecto ManuGas surge para dar solución a la necesidad de informatización que necesita la Empresa de Gas Manufacturado de Ciudad de La Habana, en el desarrollo de las principales actividades relacionadas principalmente con los procesos de Facturación y Cobro, siendo estos procesos los módulos del proyecto respectivamente. Entre los principales objetivos se encuentran: obtener un producto que permita facturar cada uno de los clientes atendiendo a los diferentes criterios de facturación y posibilidades existentes. Permitir que cada cliente pueda realizar el pago del servicio del gas en cualquier Casa Comercial, puesto que actualmente tiene que efectuar el pago en la casa comercial de su municipio. Otro de los objetivos que se pretende cumplir es el de poder obtener cuadres parciales del estado de los procesos de Facturación y Cobro, lo que permitirá mejorar la toma de decisiones y conocer el estado de la empresa. Al concluir el proyecto se instalará en la empresa y se entregará toda la documentación que avale la ejecución del mismo.

2.2 Características a probar

Las características a evaluar en un software son en dependencia del tipo de producto que sea y de lo que se pretenda probar con la prueba a realizar.

Hay algunas características que siempre se deben tener en cuenta, como por ejemplo: si la aplicación cumple con las funcionalidades requeridas desde el comienzo; si no tiene errores a la hora de ejecutarlo; si es comprensible para el usuario; si se puede usar correctamente con facilidad y si cuenta con una documentación detallada y que corresponda con la aplicación. También se pueden medir aspectos generales que tienen que ver con el código del programa y permiten conocer si dicho código está bien estructurado y es reutilizable. Por ello es importante comprobar que la aplicación sea fácil de mantener para garantizar que se mantenga activa por mucho tiempo. (10)

2.3 Plan de prueba del producto ManuGas

El documento plan de prueba para el sistema ManuGas define los siguientes objetivos:

- Identificar los elementos que se van a probar.
- Describir la estrategia que se va a seguir en el proceso de prueba.
- Listar los resultados que se obtienen como resultado del proceso de prueba.

2.4 Características del plan de prueba

El plan de prueba describe las pruebas de unidad y sistema aplicadas al producto ManuGas del proyecto Sistema de Facturación y Cobro para la empresa de Gas Manufacturado. El objetivo es probar todos los requisitos establecidos en la Especificación de Requisitos y en el Modelo de Caso de Uso, así como revisar la correcta redacción, ortografía y estructuración de los documentos. Se especifican además los roles y la responsabilidad de cada estudiante en específico. Se hace referencia al listado de requerimientos a probar, incluyendo los requisitos funcionales del sistema. También se describe el flujo de trabajo que se utilizará para la ejecución de las pruebas, teniendo en cuenta la estrategia de prueba a seguir.

Se realiza un cronograma de ejecución de las pruebas plasmando la tarea, fecha de inicio, responsable y participantes de la misma. Finalmente se evalúa el resultado de las pruebas y se registran los errores en la planilla de no conformidades para ser corregidos posteriormente por el equipo de desarrollo del proyecto.

2.5 Descripción de la Estrategia de las pruebas realizadas

La estrategia de software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo para la realización de un proceso de pruebas, como son: la planificación; el diseño de casos de prueba; la ejecución y los resultados de las pruebas. Antes de empezar a explicar cómo fue que se realizaron las pruebas, es preciso decir que éstas se centran en cada módulo individual.

Primero se revisa la documentación, para ello es necesario revisar la planilla Modelo de Caso de Uso del sistema. Los errores obtenidos después de haber realizado las pruebas se guardan en la planilla de No Conformidades, la cual es enviada a los desarrolladores de la aplicación para su posterior corrección.

2.6 Niveles de prueba seleccionados

Los niveles de pruebas seleccionados para desarrollar las pruebas a la aplicación informática ManuGas son los niveles de unidad y sistema.

El nivel de unidad es escogido porque tiene como objetivo verificar la funcionalidad y estructura de cada componente individualmente una vez que ha sido codificado. Es un proceso para probar los subprogramas, las subrutinas, los procedimientos individuales o las clases en un programa. Las pruebas de unidad son una manera de manejar los elementos de prueba combinados, puesto que se centra la atención inicialmente en unidades más pequeñas del programa. Facilita la tarea de eliminar errores (el proceso de establecer claramente y de corregir un error descubierto), puesto que, cuando se encuentra un error, se sabe que existe en un módulo particular. Finalmente, las pruebas de unidad introducen paralelismo en el proceso de pruebas del software presentándose la oportunidad de probar los múltiples módulos simultáneamente y son en gran parte orientadas a caja blanca. (7)

La prueba de sistema es la encargada de verificar que cada elemento encaja de forma adecuada y que se alcance la funcionalidad y el rendimiento del sistema total. Está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadoras.

2.7 Método de prueba seleccionado

La prueba de Caja Blanca es un tipo de prueba de software que se realiza sobre las funciones internas de un módulo, o sea, estas pruebas las hace el mismo programador una vez que realiza la implementación de alguna funcionalidad dentro del módulo.

En este trabajo de diploma, las pruebas que se realizan son más allá de las pruebas de caja blanca, es decir las de caja negra, pruebas que ejercitan los requisitos funcionales desde el exterior del módulo.

Durante el desarrollo del capítulo 1 se estudiaron los dos métodos de prueba que existen, para ello se hizo una breve descripción de su contenido. Se hizo énfasis en el Método de Caja Negra teniendo en cuenta que se basa en la aplicación de las pruebas a la interfaz del software y no al código interno como lo demuestra el Método de Caja Blanca.

Debido a lo anteriormente expuesto se selecciona el Método de Caja Negra, utilizando el criterio de partición equivalente, con el objetivo de comprobar el buen funcionamiento de la aplicación.

2.7.1 Partición Equivalente

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. El objetivo de partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases. La partición equivalente es subjetiva. Dos probadores quienes prueban un programa complejo pueden llegar a diferentes conjuntos de particiones. (7)

Dentro del nivel de sistema se utilizará la prueba de funcionalidad, que tiene como objetivo asegurar el apropiado trabajo con los requisitos funcionales. Esta prueba utiliza el método de caja negra en el cual se ejecuta cada caso de uso, flujo de caso de uso o función, usando datos válidos e inválidos, para verificar los siguientes puntos:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

2.8 Metodología de desarrollo de software seleccionada

Se seleccionó la metodología RUP porque es la que predefine el proyecto Sistema de Facturación y Cobro para la empresa de Gas Manufacturado en el desarrollo del producto Gas Manufacturado. Se tuvo en cuenta que RUP constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. No es un sistema con pasos firmemente establecidos, sino que es un conjunto de metodologías adaptables al contexto y necesidades de cada organización, por lo que ha permitido adaptarla a las necesidades que impone el producto que se desarrolla. RUP como metodología se rige por 5 procesos básicos: Adaptar al proceso, establecer prioridades, demostrar valor iterativamente, elevar el nivel de abstracción y el enfoque en la calidad. Además administra de una forma muy eficiente los requisitos y el control de cambios.

2.9 Diseño de Casos de Prueba

El diseño de caso de prueba está totalmente mediatizado por la imposibilidad de probar exhaustivamente el software. Es muy importante destacar que los diseños de casos de pruebas se realizaron según la descripción de los casos de uso del sistema propuesto por el proyecto. Por eso la idea fundamental del diseño de caso de prueba consiste en elegir algunas de éstas funcionalidades, que por sus características se consideren representativas del resto. Así se asume que si no se detectan defectos en el software al ejecutar dichos casos, se puede tener cierta confianza en que el software no tiene defectos.

La dificultad está en elegir los casos que se deben ejecutar. Una elección aleatoria no proporciona demasiada confianza en que se puedan detectar todos los defectos existentes.

2.9.1 Estructura del diseño de Caso de Prueba

1. Descripción General: Se describe el caso de uso con el que voy a trabajar y las especificaciones necesarias para el desarrollo del mismo.

2. Condiciones de ejecución: Condiciones que hay que tener en cuenta para el funcionamiento del software.

3. Secciones a probar: Este punto describe lo que realmente debe hacer el caso de uso. Para esto se debe llenar una tabla con el nombre de la sección, el escenario de la sección, la descripción de la funcionalidad y el flujo central. Se describe además las variables que serán probadas con sus valores válidos e inválidos que pueden tomar con su comportamiento en el sistema.

4. Registro de defectos y dificultades detectadas: Se registra en una tabla las no conformidades detectadas después de haber probado las variables.

2.9.2 Módulo Facturación:

Caso de Uso: Cargar TPL

Resumen: Este caso de uso comienza cuando la Tramitadora de la Casa Comercial va a cargar los datos de los clientes a los que se le va a efectuar la lectura, selecciona la ruta a la que se le va a efectuar la lectura, genera el fichero con los datos, crea el Modelo de Tramitadora y carga los datos en el TPL.

Precondiciones: Se encuentre autentificado en el sistema un usuario con el Rol de Tramitadora.

Secciones a probar en el Caso de Uso:

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**
SC 1: Cargar TPL	EC 1.1: Cargar TPL exitosamente.	El sistema muestra un listado con las rutas a facturar por el lector cobrador, luego pide la selección de la ruta y el folio por las que se va a facturar. Prepara los datos de los clientes que se van a facturar por número de Ruta y Folio y busca los datos de los clientes que pertenecen a las rutas seleccionadas. Después carga los datos de los clientes que pertenecen a la ruta seleccionada, mostrando los datos de los clientes que se van a facturar. Crea un Modelo de Tramitadora con los datos del cliente que se van a facturar e imprime el Modelo de Incidencias. El sistema imprime los datos de los clientes que se van a facturar y exporta los datos de los clientes que se van a facturar a un archivo.dbf
	EC 1.2: Cargar TPL dejando el campo de nombre en blanco.	El sistema no muestra un listado con las rutas a facturar por el lector cobrador o luego detecta que la selección de la ruta y el folio por las que se va a facturar no es correcta. Por lo que no prepara los datos de los clientes que se van a facturar por número de Ruta y Folio y no busca los datos de los clientes que pertenecen a las rutas seleccionadas. Después no carga los datos de los clientes que pertenecen a la ruta seleccionada, informando al usuario del error ocurrido durante el proceso, terminando el CU.
	EC 1.3: Cargar TPL introduciendo números en el campo de nombre.	El sistema no muestra un listado con las rutas a facturar por el lector cobrador o luego detecta que la selección de la ruta y el folio por las que se va a facturar no es correcta. Por lo que no prepara los datos de los clientes que se van a facturar por número de Ruta y Folio y no busca los datos de los clientes que pertenecen a las rutas seleccionadas. Después no carga los datos de los clientes que pertenecen a la ruta seleccionada, informando al usuario del error ocurrido durante el proceso, terminando el CU.

Tabla 1: Diseño de Caja Negra al Caso de Uso: Cargar TPL

Caso de Uso: Leer TPL

Resumen: El caso de uso comienza cuando la Tramitadora de la Casa Comercial escoge la opción de Descargar las Lecturas de los Clientes del TPL, comprueba los datos de éstas lecturas con los datos existentes en el Modelo de la Tramitadora, corrige los errores, las incidencias existentes, calcula el consumo de cada cliente y envía estos datos hacia a la UEB Comercial.

Precondiciones: Se encuentre autentificado en el sistema un usuario con el Rol de Tramitadora.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Leer TPL	EC 1.1: Leer TPL exitosamente	El sistema carga el fichero leyendo todos los datos de los clientes. Muestra un listado de todas las lecturas corregidas, marcando las que presentaron errores. Muestra además un listado de las lecturas brindando la posibilidad de corregir los datos del modelo de incidencia que son la lectura y la clave. Calcula el consumo de cada cliente leído, restando la lectura actual con la lectura del mes anterior. Guarda en la Base de Datos los datos de las lecturas y el consumo de cada cliente y muestra un listado con la lectura y el consumo de cada cliente.
	EC 1.2: No lee el TPL	El sistema encuentra incongruencias en los datos cargados y los datos existentes en el Modelo de la Tramitadora, reportando esto a la Tramitadora de la Casa Comercial. Detecta que él % de lecturas efectuadas no supera al % de lecturas esperado, reporta esto a la Tramitadora de la Casa Comercial. Detecta además que algunos de los datos corregidos están incorrectos, volviendo a pedir la entrada de estos datos.

Tabla 2: Diseño de Caja Negra al Caso de Uso: Leer TPL

Caso de Uso: Realizar Facturación

Resumen: El Caso de Uso comienza cuando el Analista selecciona la Casa Comercial que va a facturar, se actualiza los datos en la Base de Datos, selecciona la Casa Comercial que desea facturar, los criterios por los que va a facturar, se factura a cada cliente seleccionado, se corrigen los errores ocurridos en el cálculo de la Facturación, se guardan los datos, se imprimen la chequeras y los cobros, finaliza el Caso de Uso.

Precondiciones: Se encuentre autentificado en el sistema un usuario con el rol de Analista del Sistema.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Realizar facturación	EC 1.1: Realizar facturación exitosamente.	El sistema carga desde un fichero la Tarifa Escalonada por la que se va a facturar a los clientes, teniendo en cuenta los datos de la Casa Comercial y los de los clientes. Busca y carga el consumo desde la Base de Datos de SISMET de los clientes, teniendo en cuenta los criterios, realiza la facturación. El sistema factura a cada cliente teniendo en cuenta el consumo de éste y los datos cargados de la Tarifa Escalonada. Muestra un reporte con los datos de la Facturación de cada cliente, permitiendo que estos datos sean corregidos. El sistema guarda los datos de la facturación de los clientes en la Base de Datos de SISMET. Imprime las Chequeras y el Listado de Cobros.

Tabla 3: Diseño de Caja Negra al Caso de Uso: Realizar Facturación

Caso de Uso: Cargar Datos a Facturar

Resumen: El Caso de Uso comienza cuando el Analista selecciona la opción de Cargar los Datos que fueron enviados por la Tramitadora de la Casa Comercial, se actualiza los datos en la Base de Datos, finaliza el Caso de Uso.

Precondiciones: Se encuentre autentificado en el sistema un usuario con el rol de Analista del Sistema.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Cargar Datos a Facturar	EC 1.1: Cargar Datos a Facturar correctamente.	El sistema carga los datos de todos los clientes que se encuentran en el fichero (lectura, consumo) y actualiza en la Base de Datos de SISMET los datos de los clientes.

Tabla 4: Diseño de Caja Negra al Caso de Uso: Cargar Datos a Facturar

Caso de Uso: Exportar Datos de Lectura

Resumen: El Caso de Uso comienza cuando la Tramitadora Casa Comercial exporta los datos de las lecturas.

Precondiciones: Se encuentre autentificado en el sistema un usuario con el rol de Tramitadora Casa Comercial.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Exportar Datos de Lectura	EC 1.1: Exportar Datos de Lectura exitosamente.	El sistema exporta los datos hacia un fichero. La Tramitadora de la Casa Comercial envía el fichero con los datos al Analista del Sistema de la UEB Comercial.

Tabla 5: Diseño de Caja Negra al Caso de Uso: Exportar Datos de Lectura.

Caso de Uso: Imprimir Cobros

Resumen: Este Caso de Uso comienza cuando el Analista del Sistema manda a imprimir todos los talonarios de cobro realizados por el cobrador.

Precondiciones: Se encuentre autentificado en el sistema un usuario con el Rol Analista del Sistema.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Imprimir Cobros	EC 1.1: Imprimir Cobros correctamente.	El sistema busca los clientes que han realizado el pago para imprimir los datos necesarios y su importe. Muestra un mensaje con la opción de si desea realizar la impresión, si acepta la opción, imprimir el cobro, manda a imprimir los cobros de los clientes.
	EC 1.2: Cancelar impresión.	El sistema busca los clientes que han realizado el pago para imprimir los datos necesarios y su importe. Muestra un mensaje con la opción de si desea realizar la impresión, si no acepta la opción imprimir no puede imprimirse el cobro realizado.

Tabla 6: Diseño de Caso de Prueba al Caso de Uso: Imprimir Cobros

2.9.3 Módulo de Cobro

Caso de Uso: Buscar Cliente

Resumen: El caso de uso se inicia cuando la Cajera teclea la opción de Buscar Cliente luego de haber tecleado los datos en el sistema.

Precondiciones: Se encuentre autentificado en el sistema un usuario con el rol de Cajera.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Buscar Cliente	EC 1.1: Buscar Cliente exitosamente.	El sistema envía los datos a la Base de Datos y verifica que el cliente existe.
	EC 1.2: Dejar campos vacíos.	El sistema muestra un mensaje de error “Debe llenar el campo de búsqueda”

Tabla 7: Diseño de Caja Negra al Caso de Uso: Buscar Cliente

Caso de Uso: Comprobar las cuentas cobradas y por cobrar

Resumen: El caso de uso se inicia cuando el actor comprueba las cuentas que les han sido entregadas por los cobradores con las que tiene en el sistema.

Precondiciones: Se encuentre autentificado en el sistema un usuario con el rol Controlador Económico.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Comprobar las cuentas cobradas y por cobrar	EC 1.1 Comprobar las cuentas cobradas.	El sistema busca en la Base de Datos las cuentas que han sido cobradas. Envía las cuentas cobradas y muestra listado de las cuentas cobradas.
	EC 1.2 Comprobar las cuentas por cobrar.	El sistema no busca en la Base de Datos las cuentas que han sido cobradas, no envía ni muestra el listado de las cuentas cobradas.

Tabla 8: Diseño de Caja Negra al Caso de Uso: Comprobar las cuentas cobradas y por cobrar

Conclusiones parciales

En este capítulo se diseñaron los casos de pruebas pertinentes a la aplicación de ManuGas, para ello se describieron las características principales del sistema a probar. Con estas pruebas se hizo una revisión detallada de los casos de uso y de la documentación generada durante todo el ciclo de desarrollo del proyecto. Quedó elaborado el plan de prueba y se argumentaron los niveles y métodos de pruebas utilizados para comprobar el buen funcionamiento del sistema, se argumentó sobre la metodología de desarrollo de software seleccionada, además se describió la estructura de la planilla del diseño de caso de prueba y la estrategia a seguir para dicha aplicación, teniendo en cuenta que los errores encontrados serán documentados y evaluados en el próximo capítulo.

Capítulo 3: Ejecución y Evaluación de las pruebas al software

Introducción

Una vez realizado el diseño de caso de prueba al producto ManuGas, se pasa a la aplicación de las pruebas. Precisamente en este capítulo se llevará a cabo un análisis de los resultados obtenidos del producto. Para ello es necesario registrar los errores encontrados al aplicar los diseños de caso de prueba desarrollados para las funcionalidades del sistema y así enviarlos al equipo de desarrollo para que sean corregidos antes de ser entregado el producto al usuario final.

3.1 Aplicación de la Lista de Chequeo a los Diseños de Caso de Prueba

Existen varias listas de chequeo, una para cada artefacto de tipo de documentación. Los diseños de casos de prueba son un artefacto de este tipo por lo cual, para revisarlos es necesario usar la lista de chequeo para casos de pruebas.

La lista de chequeo es aplicada a los diseños de caso de prueba para comprobar su correcta implementación, es decir, es una prueba aplicada a los casos de prueba para asegurar que están bien diseñados. Durante la aplicación de la lista de chequeo a dichos documentos se pudo comprobar que todos arrojaron los mismos resultados.

La diferencia que existe en aplicar las pruebas o la lista de chequeo a los diseños de caso de prueba y las pruebas aplicadas a la aplicación como tal, en este caso al sistema de Gas Manufacturado, es precisamente que los errores detectados en los diseños de casos de pruebas los corrige el propio diseñador porque fue el que los diseñó, mientras que los errores detectados en la aplicación son enviados al equipo de desarrollo para su posterior corrección.

3.2 Estructura de las No Conformidades

En la planilla de No Conformidades se describen los aspectos a tener en cuenta a la hora de analizar los resultados de las pruebas. Este punto cuenta con dos aspectos:

1. Aspectos Generales.
2. Elementos probados.

1. Tablas de No Conformidades Detectadas.

Dicha tabla presenta 7 aspectos a llenar, estos son:

Elemento: Elementos que se revisan.

Nro.: Número de la No Conformidad.

No Conformidad: Se describe el error como tal.

Aspecto correspondiente: Es la parte específica donde se encontró dicho error.

Etapa de Detección: Etapa de Detección en la que se encontró el error.

Clasificación: Las No Conformidades se clasifican en S (significativa), N (no significativa) y R (recomendaciones).

Estado de la No Conformidad: Se coloca el estado de la NC (No Conformidad) y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó. RA (Resuelta), PD (Pendiente) y NP (No procede).

Respuesta del equipo de desarrollo: Se llena a partir de la última iteración, es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado y en el caso de no proceder la no conformidad, explica por qué.

2. **Anexos: Si existen se colocan fotos o algo que tenga que ver con la No Conformidad.**

3.2.1 No Conformidades detectadas después de aplicar las pruebas al producto ManuGas

Las No Conformidades son defectos encontrados durante la etapa de ejecución de las pruebas. Estos errores detectados son de gran importancia para el software, puesto que asegura un alto nivel de calidad en el producto, partiendo del criterio de que mientras menos defectos se encuentren en el software más importante se vuelven en el mercado mundial.

A continuación se representa mediante una tabla las No Conformidades detectadas por cada Caso de Uso.

Funcionalidad: Cargar TPL

Elem	No	No Conform.	Ubicac. No Conform.	Etapa de Detec.	Sig.	No Sig.	Recomendac.	Estado NC	Resp.Equipo Desarrollo
Aplic.	1	El nombre de la funcionalidad Crear DBF implementada en el sistema no coincide con el nombre Cargar TPL descrito en el modelo de Caso de Uso del sistema.	Se ubica en el rol de Tramitadora Facturación. Crear DBF.	1ra Iteración	X			PD: 03-05-2010.	

Aplic.	2	En la descripción del Caso de Uso se plantea que se debe de introducir el nombre del fichero que se va crear con los datos de los clientes a facturar y se da clic en el botón aceptar, mientras que en la aplicación no se especifica esa opción.	Se ubica en el rol de Tramitadora Facturación. Crear DBF.	1ra Iteración	X			PD: 03-05-2010.	
Aplic.	3	Para seleccionar cualquier dato en la aplicación, hay que presionar varias veces	Se ubica en el rol de Tramitadora Facturación. Crear DBF.	1ra Iteración	X			PD: 03-05-2010.	

		dando clic.							
Aplic.	4	En la descripción del caso de uso se especifica la opción de Imprimir el Modelo de Incidencias mientras que en la aplicación no existe la opción.	Se ubica en el rol de tramitadora. Facturación. Crear DBF.	1ra Iteración	X			PD: 03-05-2010.	

Tabla 9: Resultado de la sección Cargar TPL

Funcionalidad: Leer TPL

Elem	No	No Conform	Ubicac. No Conform	Etapa de Detec.	Sig.	No Sig.	Recomendac.	Estado NC	Resp. Equipo Desarrollo
Aplic.	1	El nombre de la funcionalidad Leer DBF implementada en el sistema	Se ubica en el rol de tramitadora, en el menú desplegable de	1ra Iteración	X			PD: 03-05-2010.	

		no coincide con el nombre Leer TPL descrito en el modelo de Caso de Uso del sistema.	Facturación. Leer DBF.						
Aplic.	2	El nombre del Botón Browser implementado en la aplicación no coincide con el nombre del botón Examinar descrito en el modelo de caso de uso del sistema.	Se ubica en el rol de tramitadora. Menú desplegable de Facturación. Leer DBF.	1ra Iteración	X			PD: 03-05-2010.	
Aplic.	3	El nombre del botón abrir implementado en la aplicación no coincide con el nombre del botón aceptar descrito en el modelo de		1ra Iteración	X			PD: 03-05-2010.	

		caso de uso del sistema.							
Aplic	4	En la Base de Datos no aparece ningún fichero con extensión DBF por lo que se hace imposible probar la funcionalidad.	Se ubica en el rol de Tramitadora, en el menú desplegable de Facturación. Leer DBF.	1ra Iteración	X			PD: 03-05-2010.	

Tabla 10: Resultado de la sección Leer TPL

Funcionalidad: Realizar Facturación

Elem	No	No Conform	Ubicac. No Conform.	Etapa de Detec.	Sig.	No Sig.	Recomendac	Estado NC	Resp.Equipo Desarrollo
Aplic.	1	El nombre de la funcionalidad Facturación, implementada en el sistema no coincide con Realizar Facturación descrita en el modelo de Caso de Uso del sistema	Se ubica en el menú desplegable de Facturación. Facturar.	1ra Iteración	X			PD: 03-05-2010.	

Aplic.	2	El nombre del Botón Facturar implementado en la aplicación no coincide con el nombre del botón aceptar descrito en el modelo de caso de uso del sistema.	Se ubica en el menú desplegable de Facturación. Facturar. Escoger cualquier criterio.	1ra Iteración	X			PD: 03-05-2010.	
Aplic.	3	En cualquiera de los criterios implementados, la aplicación acepta cualquier dato de entrada que no se encuentre en la Base de Dato.		1ra Iteración	X			PD: 03-05-2010.	
Aplic.	4	En ocasiones cuando se comienza a facturar se demora demasiado.		1ra Iteración	X			PD: 03-05-2010.	

Tabla 11: Resultado de la sección Realizar Facturación

Funcionalidad: Cargar Datos a Facturar

Elem	No	No Conform	Ubicac. No Conform	Etapa Detec.	Sig.	No Sig.	Recomendac.	Estado NC	Resp.Equipo Desarrollo
Aplic.	1	El nombre de la funcionalidad Cargar Lectura implementada en el sistema no coincide con Cargar Datos a facturar descrita en el modelo de Caso de Uso.	Se ubica en el menú desplegable de Facturación Cargar Lectura.	1ra Iteración	X			PD: 30-04-2010.	
Aplic.	2	El nombre del botón Browser implementado en la aplicación no coincide con el nombre del botón Examinar descrito en el modelo de caso de uso del sistema.	Se ubica en el menú desplegable de Facturación Cargar Lectura.	1ra Iteración	X			PD: 30-04-2010.	
Aplic.	3	El nombre del botón abrir implementado en la aplicación	Se ubica en el menú desplegable de Facturación	1ra Iteración	X			PD: 03-05-2010.	

		no coincide con el botón aceptar descrito en el modelo de caso de uso del sistema.	Cargar Lectura. Browser.						
Aplic	4	En la Base de datos no existe ningún archivo con extensión YML por lo que se hace imposible probar la funcionalidad.	Se ubica en el menú desplegable de Facturación Cargar Lectura.	1ra Iteración	X			PD: 03-05-2010.	

Tabla 12: Resultado de la sección Cargar Datos a Facturar

Funcionalidad: Exportar Datos de Lectura

Elem	No	No Conform	Ubicac. No Conform.	Etapa Detec.	Sig.	No Sig.	Recomendac.	Estado NC	Resp.Equipo Desarrollo
Aplic.	1	El nombre de la funcionalidad Exportar Lectura implementada en el sistema no coincide con el nombre Exportar datos de Lectura descrito en el modelo de Caso	Se ubica en el rol de Tramitadora Facturación. Exportar Lectura.	1ra Iteración	X			PD: 03-05-2010.	

		de Uso del sistema.							
Aplic.	2	En la descripción del caso de uso del sistema se específica que la Tramitadora de la Casa Comercial escoge la opción de exportar la lectura y el consumo de cada cliente hacia un fichero mientras que en la aplicación lo que se selecciona es la fecha de inicio y fin de las lecturas.	Se ubica en el rol de tramitadora. Facturación. Exportar Lecturas.	1ra Iteración	X			PD: 03-05-2010.	
Aplic.	3	La aplicación no exporta los datos, las fechas seleccionadas. Están inactivas.	Se ubica en el rol de Tramitadora. Facturación. Exportar Lecturas.	1ra Iteración	X			PD: 03-05-2010.	

Tabla 13: Resultado de la sección Exportar Datos de Lectura

Funcionalidad: Generar Recibo de Cobro

Elem	No	No Conform	Ubicac. No Conform.	Etapa Detec.	Sig.	No Sig.	Recomendac	Estado NC	Resp.Equipo Desarrollo
Aplic.	1	El nombre de la funcionalidad Generar Recibo de Cobro implementada en el sistema no coincide con Imprimir Cobro descrito en el modelo de Caso de Uso del sistema.	Se ubica en el menú de Facturación Generar Recibo de Cobro.	1ra Iteración	X			PD: 03-05-2010.	
Aplic.	2	El flujo central del diseño de caso de uso de Imprimir Cobro indica que el actor debe escoger el Talonario de Cobros a imprimir mientras que en la aplicación lo que se debe escoger es la Casa	Se ubica en Facturación Generar Recibo de Cobro.	1ra Iteración	X			PD: 03-05-2010.	

		Comercial a la que se le va a Generar el Reporte.							
Aplic	3	En la funcionalidad Generar Recibo de Cobros, cuando se selecciona la Casa Comercial no se Genera ningún Reporte.	Se ubica en Facturación Generar Recibo de Cobro.	1ra Iteración	X			PD: 03-05-2010.	

Tabla 14: Resultado de la sección Generar Recibo de Cobro

Funcionalidad: Buscar Cliente

Elem	No	No Conform	Ubicac. No Conform.	Etapa Detec.	Sig.	No Sig.	Recomendac.	Estado NC	Resp.Equipo Desarrollo
Aplic	1	El nombre Registrar Cobro implementado en la aplicación no coincide con el nombre Buscar cliente descrito en el caso de uso del sistema.	Se ubica en el rol de Tramitadora del menú desplegable de Cobro. Registrar Cobro.	1ra Iteración	X			PD: 03-05-2010.	

Aplic	2	En la pestaña Consumidor en los campos de Nombre, Calle y Número acepta cualquier variable.	Se ubica en el rol de Tramitadora del menú desplegable de Cobro. Registrar Cobro.	1ra Iteración	X			PD: 03-05-2010.	

Tabla 15: Resultado de la sección Buscar Cliente

Funcionalidad: Cuentas por cobrar

Elem	No	No Conform	Ubicac. No Conform.	Etapa Detec.	Sig.	No Sig.	Recomendac.	Estado NC	Resp.Equipo Desarrollo
Aplic	1	El nombre de la funcionalidad Cuentas por Cobrar implementada en el sistema no coincide con el nombre Comprobar cuentas cobradas y por cobrar descrito en el modelo de Caso de Uso del sistema.	Se ubica en el menú desplegable de Reportes. Cuentas por cobrar.	1ra Iteración	X			PD: 03-05-2010.	

Aplic	2	El flujo central del diseño de caso de uso de Comprobar las Cuentas Cobradas y por Cobrar indica que el actor debe seleccionar la opción de comparar las cuentas cobradas mientras que en la aplicación lo que se debe escoger es la casa comercial a la que se le va a Generar el Reporte.	Se ubica en el menú desplegable de Reporte. Cuentas por Cobrar.	1ra Iteración	X			PD: 03-05-2010.	
Aplic	3	En la funcionalidad Cuentas por Cobrar, cuando se selecciona la Casa Comercial no se Genera ningún Reporte.	Se ubica en el menú desplegable de Reporte. Cuentas por Cobrar.	1ra Iteración	X			PD: 03-05-2010.	

Tabla 16: Resultado de la sección Cuentas por cobrar

3.3 Resultados Generales

Durante el proceso de ejecución de las pruebas se detectaron en sentido general una serie de errores en ambos módulos, denominados No Conformidades. Del total de errores detectados, 8 se enmarcaron principalmente en que el nombre de la funcionalidad implementada en la aplicación no coincide con el nombre del Caso de Uso descrito en el Modelo de Caso de Uso del Sistema, 5 de ellos no coinciden los nombres de los botones implementados con los descritos en el Modelo de Caso de Uso. Además existe problema con la descripción de los Casos de Uso, pues hay funcionalidades que no están descritas previamente o que ya están descritas pero no implementadas, detectándose 4 errores de este tipo. Producto a esto surgen los problemas en la aplicación, provocando que algunas funcionalidades implementadas en el sistema carguen lentamente, en este aspecto se detectaron 10 No Conformidades. A continuación se presenta una figura en la que se muestra la cantidad en porciento de No Conformidades de cada tipo.

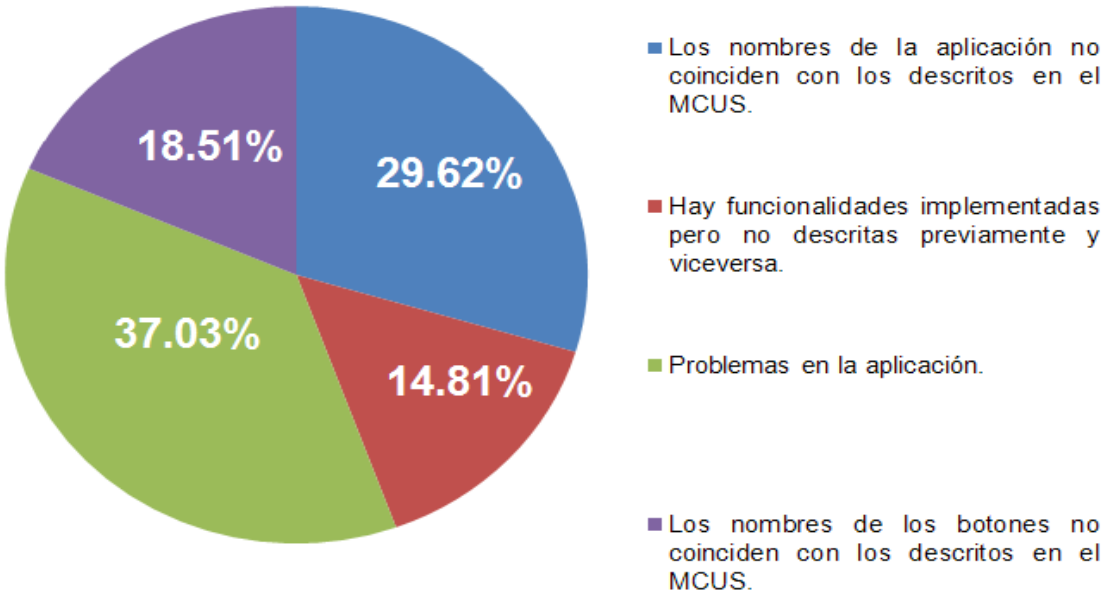

Figura 1: Cantidad de No Conformidades de cada tipo

De los 8 diseños de caso de prueba, 6 pertenecen al módulo de Facturación y 2 al módulo de Cobro. Para un total de 27 No Conformidades detectadas en ambos módulos. Se pudo comprobar que en el módulo de Facturación la mayor parte de los errores se detectaron al aplicar los diseños de caso de prueba Cargar TPL, Leer TPL, Realizar Facturación y Cargar Datos a Facturar presentando 4 No Conformidades cada uno, mientras que en el módulo de Cobro la mayor parte de los errores se detectaron al aplicar los diseño de caso de prueba Generar Recibo de Cobro y Cuentas por Cobrar. En la siguiente figura se detalla con mejor claridad la cantidad de no conformidades detectadas por cada caso de uso.

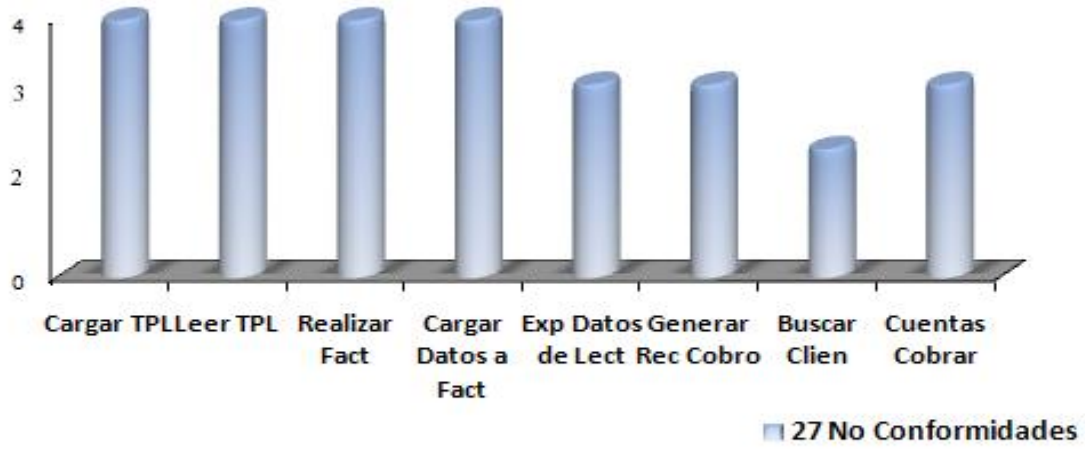

Figura 2: Cantidad de No Conformidades por Caso de Uso

Conclusiones parciales

En este capítulo se llevó a cabo el análisis de los resultados obtenidos luego de aplicar las pruebas al producto ManuGas. Se argumentó la importancia de aplicar la lista de chequeo a los diseños de caso de pruebas. Se explicó la estructura de la planilla de No Conformidades. Se registraron los errores detectados y se hizo un resumen sobre los resultados generales alcanzados durante la ejecución de las pruebas. En sentido general los resultados de las pruebas fueron muy productivos y satisfactorios, debido a que éstos arrojaron gran cantidad de no conformidades, lo que en conjunto con la colaboración del equipo de desarrollo del proyecto serán corregidos posteriormente, siendo muy importante para la aceptación final del producto, pues la satisfacción del cliente es el principal objetivo de las pruebas y de todo el proceso de calidad.

Conclusiones Generales

Debido a la necesidad que existe de tener software confiable y seguro, se hace imprescindible llevar a cabo una estrategia para asegurar la calidad de los mismos. La industria del software ha crecido inmensamente y junto con ella ha aumentado la calidad de sus productos, dado que mientras menos defectos tenga y más seguro sea un software, más importante se vuelve en el mercado mundial.

Para lograr un alto nivel de calidad en el producto se hizo necesario realizar un proceso de prueba basándose en comprobaciones específicas en cada caso y en cada fase de desarrollo del producto.

Al planificarse y aplicarse las pruebas pertinentes a la aplicación ManuGas, se llegó a las siguientes conclusiones:

- Se logró alcanzar un alto nivel de conocimiento en cuanto a los tipos de pruebas y métodos aplicados al producto ManuGas.
- Durante el proceso de pruebas se crearon los artefactos: Diseño de Caso de Prueba, Plan de prueba y la Planilla de No Conformidades.
- Se diseñaron casos de prueba para verificar el buen funcionamiento de la aplicación.
- Se realizaron las pruebas funcionales empleando el método de Caja Negra con vistas a lograr la obtención del mayor número de defectos en el software.
- Se logró documentar los errores detectados en la planilla de No Conformidades, los cuales serán corregidos posteriormente por el equipo de desarrollo del proyecto.

Las pruebas de software son un punto clave para la determinación del nivel de calidad de cualquier aplicación, es por eso que definirlas y aplicarlas constituye una necesidad para la industria cubana del software.

Recomendaciones

Debido a la importancia que lleva el proceso de prueba para la corrección de errores en una aplicación. Se recomienda:

- Aplicar las pruebas relacionadas con el método de Caja Blanca a la aplicación ManuGas, ya que esto aseguraría de forma general la calidad del producto.
- Que los errores detectados sean corregidos por el equipo de desarrollo antes de ser entregado el producto al usuario final.

Referencias Bibliográficas

1. *Organization for International Standardization.* s.l. : ISO 8402, 1994.

2. **Pressman, Roger.** *Ingeniería del Software_ un enfoque práctico.* 1993.

3. **Cueva, Juan Manuel.** *Calidad del Software.* 1999.

4. *Tema 9. Pruebas del software.*

5. **Ivar Jacobson, Grady Booch,James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

6. Analisis factores que determinan la Calidad del software. *Analisis factores que determinan la Calidad del software.* [En línea] [Citado el: 15 de 01 de 2010.] http://www.mitecnologico.com/Main/AnalisisFactoresDeterminanCalidadSoftware).

7. **Arquisoft, Grupo.** Fundamentos de las Pruebas de Software. *Fundamentos de las Pruebas de Software.* [En línea] 2007. [Citado el: 23 de 02 de 2010.] http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node4.html.

8. **yusmilaidy Causse Ascanio, Surima Ge Pérez.** *Propuesta de implementación de CMMI en el área de proceso: Aseguramiento de la Calidad de los Procesos y Productos, para los proyectos productivos de la Facultad 2.* Ciudad Habana : s.n., 2008.

9. **Isabel Blank, Larissa Herrera,Miguel Ortiz.** *Pruebas de Funcionalidad.* 2005.

10. **López, Yisel Tornés.** *Diseño y aplicación de pruebas al producto PRIMICIA, Plataforma de Televisión Informativa.* Ciudad Habana : s.n., 2009.

11. **(RUP), Rational Unified Process.** *Ayuda del RUP (Rational Unified Process).* 2006.

12. **wirwin.** Ejecución Estratégica. *Ejecución Estratégica.* [En línea] 10 de 06 de 2009. [Citado el: 26 de 02 de 2010.] http://ejecucion.wordpress.com/2009/06/10/metodologia-scrum-para-la-direccion-de-proyectos-informaticos/.

13. Metodología XP VS Metodología RUP. *Metodología XP VS Metodología RUP.* [En línea] abril de 2008. [Citado el: 26 de 02 de 2010.] http://metodologiaxpvsmetodologiarup.blogspot.com/.

14. **CHACÓN, JULIO CÉSAR RUEDA.** *APLICACIÓN DE LA METODOLOGÍA RUP PARA EL.* Guatemala : s.n., 2006.

15. **Fowler, Martin.** La Nueva Metodología. *La Nueva Metodología.* [En línea] marzo/abril de 2003. [Citado el: 26 de 02 de 2010.] http://www.programacionextrema.org/articulos/newMethodology.es.html#tth_sEc5.

Bibliografía Consultada

1. Organization for International Standardization. s.l. : ISO 8402, 1994.

2. **Pressman, Roger.** Ingeniería del Software_ un enfoque práctico. 1993.

3. **Cueva, Juan Manuel.** Calidad del Software. 1999.

4. Tema 9. Pruebas del software.

5. **Ivar Jacobson, Grady Booch,James Rumbaugh.** El Proceso Unificado de Desarrollo de Software. Madrid : s.n., 2000.

6. Analisis factores que determinan la Calidad del software. Analisis factores que determinan la Calidad del software. [En línea] [Citado el: 15 de 01 de 2010.] http://www.mitecnologico.com/Main/AnalisisFactoresDeterminanCalidadSoftware).

7. **Arquisoft, Grupo.** Fundamentos de las Pruebas de Software. Fundamentos de las Pruebas de Software. [En línea] 2007. [Citado el: 23 de 02 de 2010.] http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node4.html.

8. **yusmilaidy Causse Ascanio, Surima Ge Pérez.** Propuesta de implementación de CMMI en el área de proceso: Aseguramiento de la Calidad de los Procesos y Productos, para los proyectos productivos de la Facultad 2. Ciudad Habana : s.n., 2008.

9. **Isabel Blank, Larissa Herrera,Miguel Ortiz.** Pruebas de Funcionalidad. 2005.

10. **López, Yisel Tornés.** Diseño y aplicación de pruebas al producto PRIMICIA, Plataforma de Televisión Informativa. Ciudad Habana : s.n., 2009.

11. **(RUP), Rational Unified Process.** Ayuda del RUP (Rational Unified Process). 2006.

12. **wirwin.** Ejecución Estratégica. Ejecución Estratégica. [En línea] 10 de 06 de 2009. [Citado el: 26 de 02 de 2010.] http://ejecucion.wordpress.com/2009/06/10/metodologia-scrum-para-la-direccion-de-proyectos-informaticos/.

13. Metodología XP VS Metodología RUP. Metodología XP VS Metodología RUP. [En línea] abril de 2008. [Citado el: 26 de 02 de 2010.] http://metodologiaxpvsmetodologiarup.blogspot.com/.

14. **CHACÓN, JULIO CÉSAR RUEDA.** APLICACIÓN DE LA METODOLOGÍA RUP PARA EL. Guatemala : s.n., 2006.

15. **Fowler, Martin.** La Nueva Metodología. La Nueva Metodología. [En línea] marzo/abril de 2003. [Citado el: 26 de 02 de 2010.] http://www.programacionextrema.org/articulos/newMethodology.es.html#tth_sEc5.

Apéndices

Glosario de Abreviaturas

AVL: Análisis de Valores Límites.

CaliSoft: Centro para la Excelencia en el Desarrollo de Proyectos Tecnológicos de la UCI.

GEySED: Centro de Geoinformática y Señales Digitales.

IDC: Informe Diario de Caja.

ManuGas: Gas Manufacturado.

MCUS: Modelo de Caso de Uso del Sistema.

NC: No Conformidad.

NP: No Procede.

PD: Pendiente.

RA: Resuelta.

RUP: Proceso Unificado de Modelado.

SISMET: Sistema de Metrado.

TPL: Dispositivo que se utiliza para almacenar los datos de los clientes con las lecturas de cada uno.

UEB: Unidad Empresarial de Base.

UML: Lenguaje Unificado de Modelado.

XP: Programación Extrema.

Apéndice

Glosario de Términos

Aplicación: Una aplicación es un sistema informático que los usuarios utilizan accediendo a un servidor web a través de internet o intranet.

Artefactos: Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables. (2)

Cliente: Individuo que abre espacios y compromete recursos (de tiempo, económicos, de identidad) para interactuar con otro individuo. Es quien recibe beneficios de exigencias impuestas.

Plantillas: Una plantilla es una guía que permite construir un diseño o un esquema para que quien lo utilice se asesore de ella y guarde cierta información que le resulte necesaria documentar.

Proceso: Un proceso es una definición del conjunto completo de actividades necesarias para transformar los requisitos de usuarios en un producto. Un proceso es una plantilla para crear proyectos. (2)

Producto: Desde el punto de vista de un ingeniero de software el producto obtenido son los programas, documentos y los datos que configuran el software de computadora. Pero desde el punto de vista de los usuarios es la información resultante que hace de algún modo el mundo mejor a los usuarios. (2)

Proyecto: Elemento organizativo a través del cual se gestiona el desarrollo de software. El resultado de un proyecto es una versión de un producto. (2)

Servicios: Conjunto de actividades que buscan responder a necesidades de un cliente. Un servicio es el resultado de llevar a cabo necesariamente al menos una actividad en la interfaz, entre el proveedor y el cliente.

Software: Producto que diseñan y construyen los ingenieros de software. Esto abarca programas que se ejecutan dentro de una computadora de cualquier tamaño y arquitectura, documentos que contienen formularios virtuales e impresos y datos que combinan números y texto y también incluye representaciones de audio, video e imágenes. (2)