UNIVERISDAD DE LAS CIENCIAS INFORMATICAS

FACULTAD 3

DISEÑO Y APLICACIÓN DE PRUEBAS DE SOFTWARE AL SISTEMA IPC.

TRABAJO PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA.

AUTORES:

Aineris Reyes Nápoles.

Mavis Rivera Nieblas.

TUTOR:

Lic. Arismayda Dorado Risco.

CIUDAD DE LA HABANA, DICIEMBRE, 2007

Solo los necios se encuentran satisfechos y confiados con la calidad de su trabajo.

Mercedes Milá

AGRADECIMIENTOS

De Mavis:

A mi mamá, porque sin ella no sería lo que soy, por guiarme siempre y llenarme de su inmenso cariño y sensibilidad.

A mi papá, por ser un gran hombre, por enseñarme a crecerme antes las dificultades y por quererme como solo el sabe hacerlo.

A Mima, porque a pesar de no estar en vida la llevo en el corazón como la persona más perfecta que he conocido.

A mi abuela Claritza, porque a pesar de no estar en vida me enseñó a ser fuerte con su carácter y por dejarme ser su misirrina.

A mi prima Yusy porque la quiero como la hermana que nunca tuve, y me defiende como tal.

A toda mi familia, por preocupare por mi en especial a mis tías y primos.

A Aineris mi compañera de Tesis, por darme tanto apoyo y ayuda en el desarrollo de la misma.

A todas mis amigas en especial (Dayrena, Yamilka, Yanet, Yoanna...) que siempre estuvieron conmigo en todo momento velando por mi suerte y haciendo suyo cada instante de mi vida.

A todos compañeros que me ayudaron de una u otra forma compartiendo buenos y malos momentos.

A mi tutora Arismayda por apoyarnos y guiarnos en el desarrollo de la Tesis.

A todos mis profesores, en especial (Eugenia), que me ayudaron en mi formación profesional.

Agradecimientos

De Aineris:

A mi mamá, por enseñarme a dar lo mejor de mí, ser tan exigente, quererme tanto y ser la mejor madre del mundo, aquí está el resultado de tanto esfuerzo.

A mi papá, por comprenderme siempre, confiar tanto en mí y darme tantos consejos, este triunfo es tuyo.

A mi hermana del alma Sire, por ser un gran ejemplo y la mejor persona que he conocido hasta hoy.

A primo, mejor dicho hermano Arnaldo, por estar tan lejos y a la vez tan cerca en estos años, por su apoyo y comprensión.

A mi tía Angela, por estar tan cerca todo este tiempo, por sus miles llamadas, ser tan buena y quererme tanto.

A mi abuelito Lalo, y mi toda mi familia, por estar pendiente de mí todo el tiempo y ayudarme en todo.

A mi compañera de tesis Mavis, pues juntas pudimos hacer todo esto.

A mi tutora Arismayda, por su preocupación y la ayuda brindada en este tiempo.

A mis amigos del barrio (Yaima, Alain y Noly) y del PRE- universitario, por estar tan cerca todos estos años y ser más que amigos, hermanos.

A mis amigas de la universidad y de toda la vida (Karen, Adis, Yele, Susana), por compartir conmigo tantas alegrías, músicas, fiestas, teatros, y por aguantar mis pesadeces, son lo mejores amigas que alguien pueda tener, las quiero mucho.

A todos mis compañeros de aula y amigos, que estuvieron junto a mí en estos cinco años, me hicieron pasar momentos muy agradables.

A todos mis profesores por contribuir en mi formación profesional, en especial a Eugenia por guiarnos en esta difícil tarea.

A todos los que preguntaron por mis estudios alguna vez.

Dedicatoria

A mis padres y a mi abuela Mima.

Mavis.

A mis padres, mi hermana, mi primo Arnaldo y mi abuelito Lalo.

Aineris.

DECLARACIÓN DE AUTORÍA

Declaramos que somos las únicas autoras de este trabajo y autorizamos al Departamento de Calidad y Vicerrectoría de Formación de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Aineris Reyes Nápoles

_____

Mavis Rivera Nieblas

_____

Lic. Arismayda Dorado Risco

_____

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Diseño y Aplicación de pruebas de software el sistema IPC

Autor: Aineris Reyes Nápoles y Mavis Rivera Nieblas.

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

<Aquí el tutor debe expresar cualitativamente su opinión y medir (usando la escala: muy alta, alta, adecuada) entre otras las cualidades siguientes:

- Independencia
- Originalidad
- Creatividad
- Laboriosidad
- Responsabilidad>

<Además, debe evaluar la calidad científico-técnica del trabajo realizado (resultados y documento) y expresar su opinión sobre el valor de los resultados obtenidos (aplicación y beneficios) >

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de <nota>. <Además, si considera que los resultados poseen valor para ser publicados, debe expresarlo también>

Lic. Arismayda Dorado Risco.

_____ _____

Firma Fecha

RESUMEN

El desarrollo que han alcanzado las tecnologías en los últimos años ha marcado un cambio de manera significativa en el manejo de la información en varias ramas de la sociedad. Este cambio ha permitido el acceso, almacenamiento y utilización de los datos que se procesan. En Cuba, las oficinas estadísticas han optado por automatizar sus actividades, haciendo más sencillo y seguro el procesamiento de los grandes flujos de información con que trabajan a diario.

El presente trabajo se enfoca en la aplicación de la Ingeniería de pruebas, como un proceso de vital importancia, asegurando y garantizando, en gran medida, la calidad del software que se está construyendo para el Cálculo del Índice de precios al Consumidor (IPC) para la Oficina Nacional de Estadísticas. Facilitando de esta forma que los empleados de dicha oficina puedan acceder y procesar toda la información del cálculo del IPC de una manera más efectiva.

ABSTRACT

The development that technologies have acquired in the last decades, have impacted in a signicative way the handle of the information in many aspects of the society. This change has allowed the access, storage and using of the data that are processed. The Statistics Offices in Cuba have decided to automate their activities, in order to make easier and saver the processing of the great flows of information they deal every day.

The present work is focused on applying the Testing Engineering, as a vital process, to assure and guarantee the quality of the software that is being built to process the Calculation of the Indexes of the Prices to the Consumer for the National Statistics Office.

TABLA DE CONTENIDOS

Tabla de Contenidos

INTRODUCCIÓN

En las últimas dos décadas las aplicaciones de software se han hecho cada vez más importantes en las labores cotidianas de las personas y las empresas, independientemente de cualquier factor diferenciante. Con las mejoras en los medios de la Informática y las Comunicaciones, las aplicaciones que en sus inicios eran de soporte o ayuda ahora se han convertido en complejas y vitales a todo nivel, sobre todo para las empresas, donde llegan a ser críticas y soportan, en muchos casos, la mayor parte de las operaciones de negocio.

Las empresas cubanas no están exentas de este acrecentado avance, por lo que requieren de aplicaciones para mejorar y facilitar su trabajo, y con ello alcanzar su buen funcionamiento y desarrollo. Dichas aplicaciones deben constar con la información más correcta y actual en el plano que trabaja la empresa, para poder satisfacer las necesidades de los clientes.

La Oficina Nacional de Estadísticas, con su sede en Ciudad de La Habana, es la entidad encargada de archivar y procesar gran cantidad de información de muchas esferas de nuestra sociedad. Dentro de las disímiles operaciones que se procesan se encuentra el cálculo de índices de precios, de extrema importancia para analizar el comportamiento de varios indicadores económicos y sociales de nuestro país.

El Índice de Precios del Consumidor (IPC) es específicamente, una medida estadística de la evolución de los precios de los productos y servicios que consumen los hogares cubanos. El conjunto de productos y servicios se obtiene básicamente del consumo de las familias, y la importancia de cada uno de ellos en el cálculo del IPC esta determinado por dicho consumo.

Una encuesta que se realiza cada 5 o 10 años a una muestra de hogares de Cuba, permite conocer los productos y servicios que más consume la población. La ultima encuesta que se hizo data de Diciembre de1999.

Para calcular el IPC no solo es necesario tener el Período Base, que es resultado de la Encuesta Nacional. También es necesario conocer el precio de dichos productos en un momento dado, para de ahí ver la variación de los precios y lograr el IPC.

Es importante señalar las características peculiares del IPC que se calcula en Cuba, que no tienen antecedentes en ningún país en el mundo, por la pluralidad de los mercados en nuestro país.

Por tal motivo, se llegó a la conclusión de que para hallar el IPC General, es necesario hallar 4 IPC específicos, que son: Mercado Formal, Mercado Informal, Mercado Agropecuario y Mercado en divisas. Cada uno de ellos tiene un peso determinado en el IPC General. (Dígase peso al porcentaje en importancia que representa un IPC específico con respecto al General.) También se hallan específicamente 3 tipos de IPC General: General en Cuba, General en Ciudad Habana y General todas las provincias (excepto Ciudad Habana).

Todo este proceso del cálculo de los índices de precio, que se realizaba a través de una aplicación que existía para dicho fin, en la actualidad se ha visto afectado por una creciente demanda de información con el transcurso de los años, dado que desde la última encuesta realizada hasta la fecha han acontecido una serie de cambios que dificultan la realización exitosa de dicho proceso.

Debido a la importancia que tiene para la economía del país el cálculo de los índices de precios al consumidor, la Oficina Nacional de Estadísticas (ONE) se ha visto en la necesidad de contactar un equipo de desarrollo de la Universidad de Ciencias Informáticas con el propósito de construir una nueva aplicación que les permita dar solución a este problema.

Por el valor de dicha aplicación, se hace necesario el diseño y la aplicación de un conjunto de pruebas que aseguren la eliminación de los defectos que vayan surgiendo durante el ciclo de desarrollo de este software, y en alguna medida asegurar la calidad del mismo con el mínimo de costo y tiempo.

Problema:

La no utilización de la ingeniería de pruebas durante el desarrollo del Sistema IPC impide la detección de errores en el proceso de construcción del software.

Objetivo General:

Diseñar pruebas de caja negra durante el desarrollo del Sistema IPC.

Objeto de Estudio:

El proceso de pruebas, en el desarrollo de software.

Campo de Acción:

Las pruebas de caja negra durante el ciclo de desarrollo del software.

Hipótesis:

Si se aplica la ingeniería de pruebas durante el desarrollo del Sistema IPC, entonces se minimizarán los errores en el ciclo de vida del software.

Tareas de Investigación:

- ✓ Realizar un estudio de la documentación sobre las pruebas, haciendo énfasis en las técnicas de Caja Negra.
- ✓ Analizar el proceso de pruebas que propone RUP para su utilización.
- ✓ Diseñar pruebas de caja negra para el Sistema IPC.
- ✓ Aplicar pruebas de caja negra al Sistema IPC.

Acciones:

- ✓ Confeccionar el plan de prueba.
- ✓ Definir la estrategia de prueba.
- ✓ Establecer la configuración del entorno de pruebas.
- ✓ Elaborar los datos de prueba.
- ✓ Construir los procedimientos de prueba.
- ✓ Diseñar los casos de prueba.
- ✓ Evaluar los resultados de las pruebas.

El resultado que se espera con el desarrollo de este trabajo es el diseño y la aplicación de de las pruebas al sistema en construcción, para lograr que el mismo tenga la calidad requerida y cumpla todas las funcionalidades definidas por los clientes.

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

En el primer capítulo de este trabajo, **Fundamentación teórica**, se muestran los criterios y las principales tendencias que tienen algunos autores acerca de la Ingeniería de Pruebas, haciendo énfasis en todo lo relacionado con los principios, objetivos, características, niveles, estrategias, métodos, técnicas, procedimientos de pruebas, etc. Se hará además un estudio de la metodología RUP para poder utilizar el proceso de pruebas que define.

El segundo capítulo, **Diseño y Aplicación de las Pruebas de Software**, se ha de enmarcar en los resultados que se deben obtener del proyecto en específico, en el mismo se hará referencia a la aplicación en general y se planteará un plan de pruebas, los datos necesarios para hacerlas, se definirá la estrategia a seguir, la configuración del entorno en que serán realizadas, además se hará referencia a los procedimientos de pruebas, que serán la base para el diseño y la ejecución de los casos de pruebas.

En el tercer capítulo, **Análisis de los Resultados Parciales**, se hará un breve análisis de los resultados obtenidos a raíz de la aplicación de las pruebas al software y se hará un resumen de los principales errores que se encontraron durante las pruebas.

Cada capítulo comienza con una breve introducción sobre el tema abordado en el mismo y finaliza con unas conclusiones donde se exponen los aspectos más relevantes que se trataron.

Al finalizar el trabajo se exponen las conclusiones, recomendaciones, referencias bibliográficas y anexos, aportando de esta forma una mayor visión del trabajo realizado.

CAPÍTULO # 1 FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

La etapa de pruebas constituye una de las más importantes durante el ciclo de desarrollo de un producto software, puesto que con la misma se logra depurar un software desde sus inicios, para que el mismo llegue a manos de los clientes cumpliendo todas sus funcionalidades y libre de defectos.

El proceso de pruebas cuenta con actividades bien definidas como se había dicho anteriormente, comienza con, la planificación de las pruebas, luego la ejecución, el control y por último la evaluación de las mismas, todas estas actividades son llevadas a cabo por un proceso de retroalimentación, de forma tal que las cuestiones de la calidad se resuelvan de manera efectiva en tiempo y costo.

Este flujo debe comenzar desde el inicio del proyecto con la confección del plan de pruebas (donde se definen los objetivos del mismo, los recursos necesarios para las pruebas, así como las estrategias que nos guiarán en la confección de los casos de pruebas), luego se diseñan los casos de pruebas (donde se cubrirán todos los casos posibles en que el sistema pueda fallar), por último se evaluarán los resultados de dichas pruebas de manera tal que la información obtenida sirva para ir refinando el producto que se encuentra en desarrollo.

En este capítulo se hará referencia a todo lo relacionado con la Ingeniería de pruebas, los principales conceptos expuestos por varios autores, así como todo lo relacionado con las Estrategias, Niveles, Métodos, Técnicas, Procedimientos y Plan de Pruebas, entre otros, destacando en varios temas algunos criterios generales que se ha considerado importante definir. Además se hará referencia a las definiciones que propone RUP para el proceso de pruebas.

1.2 Temas relacionados con la Ingeniería de pruebas.

1.2.1 Antecedentes de las pruebas de software.

El proceso de desarrollo de software cuenta solo con algunas décadas, por lo que los caminos a transitar no están bien definidos y aún están por aplicarse las formas más adecuadas de llevarlo a cabo.

Anteriormente el proceso de pruebas de software era considerado como una actividad que desarrollaba el programador para encontrar las fallas a los productos que iba construyendo. Con el paso de los años se ha hecho evidente la importancia que tiene desarrollar productos de software empleando el mínimo de tiempo y

costo y que a su vez cuenten con la calidad requerida, por lo tanto se ha tomado como propósito principal evaluar la funcionalidad del software respecto a los requerimientos establecidos en el inicio del proyecto.

Las nuevas tendencias de las pruebas consisten en: iniciarlas antes, dentro del proyecto y capacitar a especialistas para que sean los responsables de desarrollar dicha actividad.

Actualmente las especificaciones de pruebas se realizan al mismo tiempo que el diseño del software, por lo que se propone iniciar el análisis de las pruebas junto con el análisis del software. Se plantea entonces realizar sondeos preventivos que permitan ejecutar las pruebas tan pronto el software este listo y con ello no solo descubrir los errores sino también poder evitarlos.

También en la actualidad se debe crear una conciencia acerca de la importancia que tienen las pruebas del software y dedicar a este proceso un grupo de personas que puedan integrarse a un proyecto y se conviertan en los responsables de su calidad.

En nuestros días se calcula que la fase o proceso de pruebas representa más de la mitad del coste de un programa, en vistas de que para el mismo se requiere de un tiempo similar al de la programación, lo que obviamente acarrea un alto costo económico. [1]

1.2.2 ¿Qué es una prueba de software? Definiciones.

La prueba de software involucra las operaciones del sistema bajo condiciones controladas y evaluando los resultados. Las condiciones controladas pueden ser normales o anormales. La prueba puede intencionalmente forzar al programa a producir errores en las respuestas para determinar si los sucesos ocurren cuando no tendrían que ocurrir o cuando los hechos no suceden cuando deberían suceder. La mayoría de las grandes organizaciones asumen la responsabilidad del control de calidad y prueba de software a tal medida que en la producción se incluyen desarrolladores de sistemas (analistas, programadores) y un grupo dedicado a la prueba de software para que estos grupos antes mencionados trabajen en conjunto cumpliendo el control de calidad (prevención) y la prueba de software (detección) logrando una tarea exitosa.

Definiciones:

A continuación se hará referencia a varios de los conceptos y definiciones que le han dado a las pruebas:

“Una actividad en la cual un sistema o uno de sus componentes se ejecuta en dos o más circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto”. Probar, por lo tanto, es el proceso de ejecutar un programa con el fin de encontrar errores o fallas. [2]

“La prueba del software es un elemento critico para la garantía de calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación”. [3]

“Cualquier intento de demostrar que el software tiene propiedades por debajo de la calidad requerida”. [4]

“Un conjunto de herramientas, técnicas y métodos que hacen la excelencia del desempeño de un programa. Las técnicas para encontrar problemas en un programa son extensamente variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad”. [5]

“Las pruebas consisten en la verificación dinámica del comportamiento de un programa en una serie finita de casos de pruebas”. [6]

De estas definiciones se puede concluir que las pruebas de software son una serie de actividades que se realizan con el propósito de encontrar los errores en un sistema de software, y así poner a prueba el comportamiento del mismo.

1.2.3 ¿Qué es probar?

Probar un programa es ejercitarlo con la peor intención de encontrarle fallas. La necesidad de hacer las pruebas se evidencia en que son enormes las posibilidades de que se cometan errores humanos, dada la imposibilidad de trabajar y comunicarse de manera perfecta, e incluso en ocasiones estas fallas pueden estar dadas porque cuestiones de infraestructura, o sea que el hardware, el sistema operativo o las herramientas que se utilizan durante el proceso de desarrollo del software introduce algunos defectos o permite que los que el programador introduce no sean detectados. [7]

1.3 Las pruebas en el desarrollo de un software.

1.3.1 Objetivos de las pruebas.

Como parte que es de un proceso industrial, la fase de pruebas añade valor al producto que se maneja: todos los programas tienen errores y la fase de pruebas los descubre; ahí reside su valor. El objetivo específico de la fase de pruebas es encontrar la mayor cantidad posible de errores, con un mínimo de costo y tiempo. [9]

Es frecuente encontrarse con el error de afirmar que el objetivo de esta fase es convencerse de que el programa funciona bien. En realidad ese es el objetivo propio de las fases anteriores (¿quién va a pasar a la sección de pruebas un producto que sospecha que está mal?). Cumplido ese objetivo, lo mejor posible, se pasa a pruebas. Esto no impide reconocer que el objetivo último de todo el proceso de fabricación de programas sea hacer programas que funcionen bien; pero cada fase tiene su objetivo específico, y el de las pruebas es destapar errores, y brindar un mayor nivel de confiabilidad a los productos que se generen. [9]

1.3.2 Principios de pruebas.

Para realizar una buena aplicación de los diferentes métodos de prueba que existen, se hace necesario que los ingenieros que trabajan en esta línea tengan en cuenta una serie de principios que nos ayudarán y guiarán en la realización del diseño de los casos de pruebas que se van a efectuar, a continuación se hará referencia a los más relevantes dentro los estudiados en las bibliografías:

- A todas las pruebas se les debería poder hacer un seguimiento hasta los requisitos de los clientes (trazabilidad).
- Las pruebas deberían planificarse antes de que empiecen.
- El principio de Pareto es aplicable a la prueba del software (“donde hay un defecto, hay otros”).
- Las pruebas deberían empezar por “lo pequeño” y progresar hacia “lo grande”.
- No son posibles las pruebas exhaustivas.
- Para ser más efectivas, las pruebas deberían ser conducidas por un equipo independiente.
- Se deben evitar los casos de prueba no documentados ni diseñados con cuidado.

- No deben realizarse planes de prueba suponiendo que prácticamente no hay defectos en los programas y, por tanto, dedicando pocos recursos a las pruebas. [10], [11], [12]

1.3.3 Características de las pruebas.

Hay ciertas características que son vitales para que una prueba sea calificada como idónea, entre ellas se pueden citar las siguientes:

1. Ha de tener una alta probabilidad de encontrar un fallo. Cuanto más, mejor.
2. No debe ser redundante. Si ya funciona, no lo probamos más.
3. Debe ser la "mejor de la cosecha". Si tenemos donde elegir, elegimos la mejor.
4. No debería ser ni demasiado sencilla ni demasiado compleja. Si es muy sencilla no aporta nada, si es muy compleja a lo mejor no sabemos lo que ha fallado. [13]

1.3.4 Estrategias de pruebas.

Integran técnicas de casos de prueba en una serie de pasos planeados que llevan a la construcción exitosa del software. Describe los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar estos pasos, y cuánto esfuerzo, tiempo y recursos se van a requerir.

Cualquier estrategia de prueba debe incorporar la planeación de las pruebas, el diseño de casos de prueba, la ejecución de pruebas y la agrupación y evaluación de los datos resultantes.

Una estrategia de prueba ha de ser lo suficientemente flexible para promover la creatividad y la adecuación necesaria a los sistemas de SW, y lo suficientemente rígida para promover una planeación razonable y un seguimiento administrativo a medida que el proyecto progresa.

Características genéricas de las estrategias de prueba:

a) Las pruebas se inician a nivel módulo y se van realizando hacia la integración del sistema completo (hacia fuera).

b) Diferentes técnicas de prueba son apropiadas en diferentes puntos del tiempo.

c) Las pruebas se llevan a cabo por el desarrollador del SW o por un grupo independiente de pruebas.

d) La depuración deber ser incluida en cualquier estrategia de prueba.

e) Debe incluir pruebas de bajo y alto nivel. [14]

La estrategia de pruebas suele seguir estas etapas:

Debe comenzar con pruebas a nivel de módulo y continuar hacia la integración del sistema completo y a su instalación y se culmina con la aceptación del producto por parte del cliente.

Más específicamente:

- Se comienza en la prueba de cada módulo, que normalmente la realiza el propio personal de desarrollo en su entorno.
- 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).

1.4 Elementos fundamentales del proceso de pruebas.

1.4.1 El Proceso de Prueba.

El proceso de prueba se inicia con la confección de un plan de pruebas en base a la documentación que se tenga, tanto del proyecto como del software a probar. Dicho plan es un punto de parte para el diseño de las pruebas, las cuales, una vez detalladas (especificaciones de casos y de procedimientos), se toma la configuración del software (revisada, para confirmar que se trata de la versión apropiada del programa) que

se va a probar para ejecutar sobre ella los casos. Puede ser que algunos casos de pruebas sean reejecuciones, por lo que es conveniente documentar los defectos ya detectados, aún los no corregidos. [15]

Ver ANEXO [1].

1.4.2 Niveles de Pruebas.

Prueba de Unidades

Normalmente cabe distinguir una fase informal antes de entrar en la fase de pruebas propiamente dicha. La fase informal la lleva a cabo el propio codificador en su despacho, y consiste en ir ejecutando el código para convencerse de que "básicamente, funciona". Esta fase suele consistir en pequeños ejemplos que se intentan ejecutar. Si el módulo falla, se suele utilizar un depurador para observar la evolución dinámica del sistema, localizar el fallo, y repararlo.

En lenguajes antiguos, poco rigurosos en la sintaxis y/o en la semántica de los programas, esta fase informal llega a ser muy dura, laboriosa, y susceptible de dejar pasar grandes errores sin que se note. En lenguajes modernos, con reglas estrictas, hay herramientas que permiten análisis exhaustivos de los aspectos estáticos de la semántica de los programas: tipado de las variables, ámbitos de visibilidad, parámetros de llamada a procedimientos, etc.

Hay herramientas más sofisticadas capaces de emitir "opiniones" sobre un programa y alertar de construcciones arriesgadas, de expresiones muy complicadas (que se prestan a equivocaciones), etc. A veces pueden prevenir sobre variables que pueden usarse antes de tomar algún valor (no inicializadas), variables que se cargan pero luego no se usan, y otras posibilidades que, sin ser necesariamente errores en sí mismas, sí suelen apuntar a errores de verdad.

Más adelante, cuando el módulo parece presentable, se entra en una fase de prueba sistemática. En esta etapa se empieza a buscar fallos siguiendo algún criterio para que "no se escape nada". Los criterios más habituales son los denominados de caja negra y de caja blanca. [9]

Pruebas de Integración

Las pruebas de integración se realizan durante la construcción del sistema, involucran a un número creciente de módulos y terminan probando el sistema como conjunto.

Estas pruebas se pueden plantear desde un punto de vista estructural o funcional.

Las pruebas estructurales de integración son similares a las pruebas de caja blanca; pero trabajan a un nivel conceptual superior. En lugar de referirnos a sentencias del lenguaje, nos referiremos a llamadas entre módulos. Se trata pues de identificar todos los posibles esquemas de llamadas y ejercitarlos para lograr una buena cobertura de segmentos o de ramas.

Las pruebas funcionales de integración son similares a las pruebas de caja negra. Aquí trataremos de encontrar fallos en la respuesta de un módulo cuando su operación depende de los servicios prestados por otros módulos. Según nos vamos acercando al sistema total, estas pruebas se van basando más y más en la especificación de requisitos del usuario.

Las pruebas finales de integración cubren todo el sistema y pretenden cubrir plenamente la especificación de requisitos del usuario. Además, a estas alturas ya suele estar disponible el manual de usuario, que también se utiliza para realizar pruebas hasta lograr una cobertura aceptable.

En todas estas pruebas funcionales se siguen utilizando las técnicas de partición en clases de equivalencia y análisis de casos límite (fronteras). [9]

Pruebas de Aceptación

Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, sino una vez pasada todas las pruebas de integración por parte del desarrollador.

La experiencia muestra que aún después del más cuidadoso proceso de pruebas por parte del desarrollador, quedan una serie de errores que sólo aparecen cuando el cliente se pone a usarlo. Los desarrolladores se suelen llevar las manos a la cabeza: "Pero, ¿a quién se le ocurre usar así mi programa?"

Sea como sea, el cliente siempre tiene razón. Decir que los requisitos no estaban claros, o que el manual es ambiguo puede salvar la cara; pero ciertamente no deja satisfecho al cliente.

Por estas razones, muchos desarrolladores ejercitan unas técnicas denominadas "pruebas alfa" y "pruebas beta". Las pruebas alfa consisten en invitar al cliente a que venga al entorno de desarrollo a probar el sistema. Se trabaja en un entorno controlado y el cliente siempre tiene un experto a mano para ayudarle a usar el sistema y para analizar los resultados.

Las pruebas beta vienen después de las pruebas alfa, y se desarrollan en el entorno del cliente, un entorno que está fuera de control. Aquí el cliente se queda a solas con el producto y trata de encontrarle fallas (reales o imaginarios) de los que informa al desarrollador. Las pruebas alfa y beta son habituales en productos que se van a vender a muchos clientes. La experiencia muestra que estas prácticas son muy eficaces. [9]

1.4.3 Otros tipos de pruebas:

Recorridos (walkthroughs):

Consiste en sentar alrededor de una mesa a los desarrolladores y a una serie de críticos, bajo las órdenes de un moderador que impida un recalentamiento de los ánimos. El método consiste en que los revisores se leen el programa línea a línea y piden explicaciones de todo lo que no está meridianamente claro. Puede que simplemente falte un comentario explicativo, o que detecten un error auténtico o que el código sea tan complejo de entender/explicar que más vale que se rehaga de forma más simple. Para un sistema complejo pueden hacer falta muchas sesiones.

Esta técnica es muy eficaz localizando errores de naturaleza local; pero falla estrepitosamente cuando el error deriva de la interacción entre dos partes alejadas del programa. Nótese que no se está ejecutando el programa, sólo mirándolo con lupa, y de esta forma sólo se ve en cada instante un trocito del listado. [9]

Solidez (robustness testing):

Se prueba la capacidad del sistema para salir de situaciones embarazosas provocadas por errores en el suministro de datos. Estas pruebas son importantes en sistemas con una interfaz al exterior, en particular cuando la interfaz es humana. [9]

Prestaciones (performance testing):

A veces es importante el tiempo de respuesta, u otros parámetros de gasto. Típicamente nos puede preocupar cuánto tiempo le lleva al sistema procesar tantos datos, o cuánta memoria consume, o cuánto espacio en disco utiliza, o cuántos datos transfiere por un canal de comunicaciones. Para todos estos parámetros suele ser importante conocer cómo evolucionan al variar la dimensión del problema (por ejemplo, al duplicarse el volumen de datos de entrada). [9]

Conformidad u Homologación (conformance testing):

En programas de comunicaciones es muy frecuente que, además de los requisitos específicos del programa que estamos construyendo, aparezca alguna norma más amplia a la que el programa deba atenerse. Es frecuente que organismos internacionales como ISO y el CCITT elaboren especificaciones de referencia a las que los diversos fabricantes deben atenerse para que sus ordenadores sean capaces de entenderse entre sí.

Las pruebas, de caja negra, que se le pasan a un producto para detectar discrepancias respecto a una norma de las descritas en el párrafo anterior se denominan de conformidad u homologación. Suelen realizarse en un centro especialmente acreditado al efecto y, si se pasan satisfactoriamente, el producto recibe un sello oficial que dice: "homologado". [9]

Regresión (regression testing):

Todos los sistemas sufren una evolución a lo largo de su vida activa. En cada nueva versión se supone que o bien se corrigen defectos, o se añaden nuevas funciones, o ambas cosas. En cualquier caso, una nueva versión exige una nueva pasada por las pruebas. Si éstas se han sistematizado en una fase anterior, ahora pueden volver a pasarse automáticamente, simplemente para comprobar que las modificaciones no provocan errores donde antes no los había.

El mínimo necesario para usar unas pruebas en una futura revisión del programa es una documentación muy clara.

Las pruebas de regresión son particularmente espectaculares cuando se trata de probar la interacción con un agente externo. Existen empresas que viven de comercializar productos que "graban" la ejecución de una prueba con operadores humanos para luego repetirla cuantas veces haga falta "reproduciendo la

grabación". Y, obviamente, deben monitorizar la respuesta del sistema en ambos casos, compararla, y avisar de cualquier discrepancia significativa. [9]

Mutación (mutation testing)

Es una técnica curiosa consistente en alterar ligeramente el sistema bajo pruebas (introduciendo errores) para averiguar si nuestra batería de pruebas es capaz de detectarlo. Si no, más vale introducir nuevas pruebas. Todo esto es muy laborioso y francamente artesano. [9]

1.5 Principales métodos de pruebas.

Cualquier producto ingenieril puede ser probado de dos formas:

1. Conociendo la función específica que el producto debe de realizar, las pruebas pueden ser llevadas a cabo para demostrar que cada función es completamente operacional; a esto se le llama pruebas de caja negra.

2. Conociendo el funcionamiento interno de un producto, las pruebas pueden ser llevadas a cabo para asegurar que todas las piezas encajen; esto es, que la operación interna del producto se lleve a cabo de acuerdo a las especificaciones y que todos los componentes internos hayan sido ejecutados adecuadamente; a esto se le llama pruebas de caja blanca.

1.5.1 Prueba de Caja Blanca.

Se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

1.5.1.1 Técnicas de Caja Blanca. Prueba del camino básico.

La prueba del camino básico es una técnica que permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.

2. Se calcula la complejidad ciclomática del grafo.

3. Se determina un conjunto básico de caminos independientes.

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

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Notación de Grafo de Flujo.

Para aplicar la técnica del camino básico se debe introducir una sencilla notación para la representación del flujo de control, el cual puede representarse por un Grafo de Flujo.

Cada nodo del grafo corresponde a una o más sentencias de código fuente. Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo.

Para construir el grafo se debe tener en cuenta la notación para las instrucciones como se muestra en las figuras siguientes: Figura 1.1 y Figura 1.2.

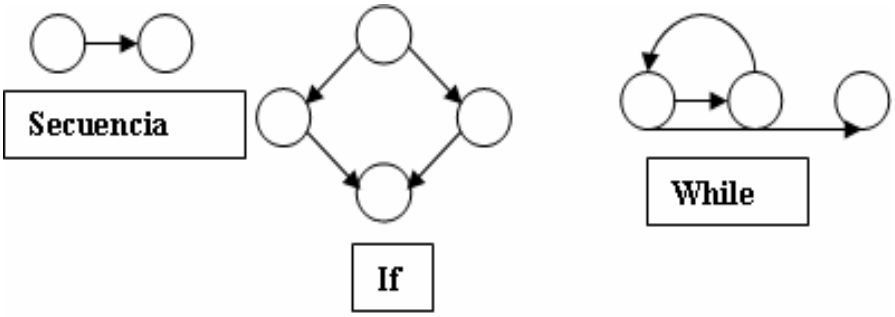


Figura 1.1 - Notación de grafos de flujo para las instrucciones: Secuenciales, If, While.

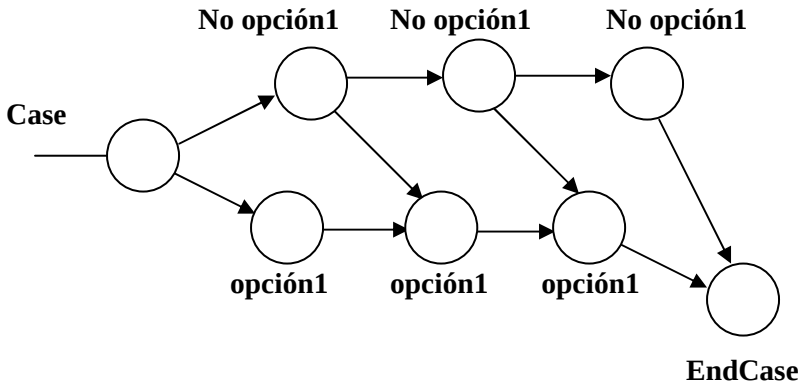


Figura 1.2 - Notación de grafos de flujo para la instrucción Case.

Un Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y nos brinda información para confirmar que el trabajo se está haciendo adecuadamente.

Los componentes son:

- Nodo.

Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hallan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.

- Aristas.

Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.

- Regiones.

Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran y la cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

Un ejemplo de representación de Grafo de Flujo es el mostrado en la Figura 1.3 en el cual aparecen sus componentes:

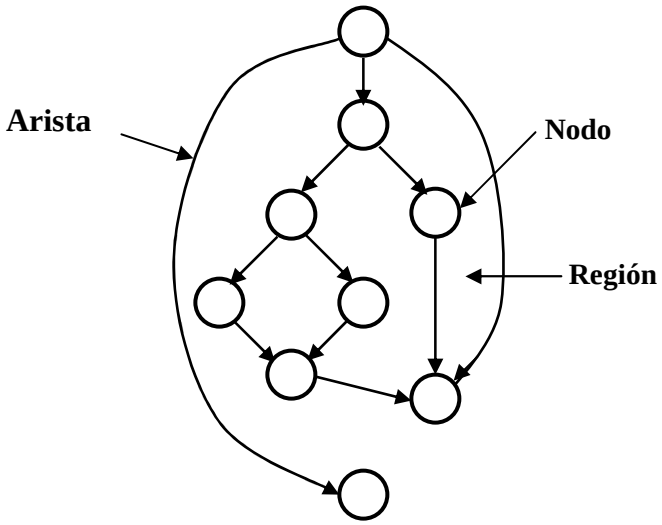
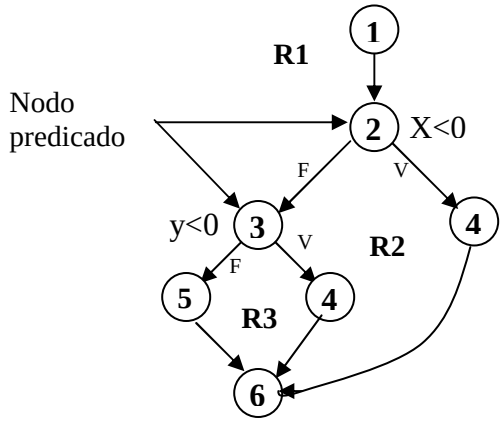


Figura 1.3 - Características de los grafos.

Cualquier representación del diseño procedimental se puede traducir a un grafo de flujo. Cuando en un diseño se encuentran condiciones compuestas (uno o más operadores AND, NAND, NOR lógicos en una sentencia condicional), la generación del grafo de flujo se hace un poco más complicada.

Ejemplo de cómo elaborar un Grafo de Flujo.

Si en un segmento de código se tiene una sentencia IF a OR b THEN entonces se crea un nodo aparte para cada una de las condiciones(a o b); cada nodo que contiene una condición se denomina nodo predicado y esta caracterizado porque dos o más aristas emergen de él, Figura 1.4.




```
PROCEDURE Imprime _ media(VAR x, y : real;)
VAR resultado : real;
resultado:=0;
IF (x < 0 OR y < 0) THEN
        WRITELN("x e y deben ser positivos");
ELSE
         resultado := (x + y)/2
         WRITELN("La media es: ", resultado);
ENDIF
END imprime_media
```

Figura 1.4 - Representación de un Grafo de Flujo.

Complejidad Ciclomática.

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente.

Ejemplo: En la Figura 1.4 un ejemplo de camino independiente sería:

Camino 1: 1-2-3-5-6.

Camino 2: 1-2-4-6.

Si se diseñan pruebas que fuercen el recorrido de esos caminos, se garantiza que se ejecute al menos una vez cada sentencia del programa y que cada condición se ejecute en sus variantes verdadera y falsa. Se debe tener en cuenta que de un mismo diseño procedimental se pueden derivar varios conjuntos básicos. [16], [12]

Formas fundamentales para calcular la complejidad:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

   La complejidad ciclomática, V (G), se define como:

   $$V(G) = A - N + 2$$

   Donde: A es el número de aristas del grafo y N es el número de nodos.

2. La complejidad ciclomática, V(G), también se define como:

   $$V(G) = P + 1$$

   Donde: P es el número de nodos predicado contenido en el grafo G.

Ejemplo: Teniendo en cuenta la Figura 1.4:

1. El grafo de flujo tiene tres regiones.
2. V (G) = 8 aristas - 7 nodos + 2 = 3.
3. V (G) = 2 nodos predicados + 1 = 3.

Por tanto la complejidad ciclomática del grafo de flujo de la Figura 1.4 es 3.

Derivación de casos de prueba.

Luego de tener elaborados los Grafos de Flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino.

Un ejemplo de derivación de casos de pruebas basándose en el grafo de la Figura 1.4 sería de la siguiente manera:

Casos de prueba para cada camino.

Camino 1: 1-2-3-5-6.

Escoger algún X y Y tales que cumpla X >= 0 AND Y >= 0.

X = 10 AND Y = 20.

Camino 2: 1-2-4-6.

Escoger algún X tal que se cumpla X < 0.

X = -15.

Luego de confeccionar los casos de prueba se ejecutan cada uno de estos y se comparan los resultados con los esperados. Una vez terminados todos los casos de prueba, se estará seguro de que todas las sentencias del programa se han ejecutado por lo menos una vez.

Es importante considerar que algunos caminos no se pueden probar de forma aislada. O sea, la combinación de datos requeridos para recorrer el camino no se puede obtener con el flujo normal del programa. En tales casos, estos caminos se prueban como parte de otra prueba de camino.

1.5.2 Prueba de Caja Negra.

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. Para realizar estas pruebas existen diferentes técnicas entre las que se encuentran:

1. Técnica Partición de Equivalencia: Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

2. Técnica Análisis de Valores Límites: Esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

3. Técnica de Grafos de Causa-Efecto: Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

De estas técnicas se hará referencia a las dos primeras pues serán las utilizadas para el diseño de casos de pruebas muy efectivos.

1.5.2.1 Técnicas de Caja Negra. Partición equivalente.

Esta técnica utiliza las cualidades que definen un buen caso de prueba de la siguiente manera:

- Cada caso debe cubrir el máximo número de entradas.
- Debe tratarse el dominio de valores de entrada dividido en un número finito de clases de equivalencia que cumplan la siguiente propiedad: la prueba de un valor representativo de una clase permite suponer "razonablemente" que el resultado obtenido (existan defectos o no) será el mismo que el obtenido probando cualquier otro valor de la clase.

El método de diseño de casos consiste entonces en:

- Identificación de clases de equivalencia.
- Creación de los casos de prueba correspondientes.

Para identificar las posibles clases de equivalencia de un programa a partir de su especificación se deben seguir los siguientes pasos:

1. Identificación de las condiciones de las entradas del programa, es decir, restricciones de formato o contenido de los datos de entrada.
2. A partir de ellas, se identifican clases de equivalencia que pueden ser:
    - De datos válidos.
    - De datos no válidos o erróneos.

   La identificación de las clases se realiza basándose en el principio de igualdad de tratamiento: todos los valores de la clase deben ser tratados de la misma manera por el programa.
3. Existen algunas reglas que ayudan a identificar clases:

- Si se especifica un rango de valores para los datos de entrada, se creará una clase válida y dos clases no válidas.
- Si se especifica un número de valores, se creará una clase válida y dos no válidas.

- Si se especifica una situación del tipo “debe ser” o booleana, se identifican una clase válida y una no válida.
- Si se especifica un conjunto de valores admitidos y se sabe que el programa trata de forma diferente cada uno de ellos, se identifica una clase válida por cada valor y una no válida
- En cualquier caso, si se sospecha que ciertos elementos de una clase no se tratan igual que el resto de la misma, deben dividirse en clases menores.

La aplicación de estas reglas para la derivación de clases de equivalencia permite desarrollar los casos de prueba para cada elemento de datos del dominio de entrada. La división en clases deberían realizarla personas independientes al proceso de desarrollo del programa, ya que, si lo hace la persona que preparó la especificación o diseñó el software, la existencia de algunas clases (en concreto, las no consideradas en el tratamiento) no será, probablemente, reconocida.

El último paso del método es el uso de las clases de equivalencia para identificar los casos de prueba correspondientes. Este proceso consta de las siguientes fases:

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

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

1.5.2.2 Técnicas de Caja Negra. Análisis de valores límite (AVL).

Mediante la experiencia se ha podido constatar que los casos de prueba que exploran las condiciones límite de un programa producen un mejor resultado para la detección de defectos, es decir, es más probable que

los defectos del software se acumulen en estas condiciones. Podemos definir las condiciones límite como las situaciones que se hallan directamente arriba, abajo y en los márgenes de las clases de equivalencia.

El análisis de valores límite es un técnica de diseño de casos que complementa a la de particiones de equivalencia. Las diferencias entre ambas son las siguientes:

1. Más que elegir "cualquier" elemento como representativo de una clase de equivalencia, se requiere la selección de uno o más elementos tal que los márgenes se sometan a prueba.

2. Más que concentrarse únicamente en el dominio de entrada (condiciones de entrada), los casos de prueba se generan considerando también el espacio de salida.

El proceso de selección de casos es también heurístico, aunque existen ciertas reglas orientativas. Aunque parezca que el AVL es simple de usar (a la vista de las reglas), su aplicación tiene múltiples matices que requieren un gran cuidado a la hora de diseñar las pruebas. Las reglas para identificar clases son las siguientes:

1. Si una condición de entrada especifica un rango de valores se deben generar casos válidos para los extremos del rango y casos no válidos para situaciones justo más allá de los extremos.

2. Si la condición de entrada especifica un número de valores ("el fichero de entrada tendrá de 1 a 255 registros"), hay que escribir casos para los números máximo, mínimo, uno más del máximo y uno menos del mínimo de valores (0, 1, 255 y 256 registros).

3. Usar la regla 1 para la condición de salida ("el descuento máximo aplicable en compra al contado será el 50%, el mínimo será el 6%"). Se escribirán casos para intentar obtener descuentos de 5,99%, 6%, 50% y 50,01 %.

4. Usar la regla 2 para cada condición de salida ("el programa puede mostrar de 1 a 4 listados"). Se escriben casos para intentar generar 0, 1, 4 y 5 listados.

En esta regla, como en la 3, debe recordarse que:

• Los valores límite de entrada no generan necesariamente los valores límite de salida (recuérdese la función seno, por ejemplo).

• No siempre se pueden generar resultados fuera del rango de salida (pero es interesante considerarlo).

5. Si la entrada o la salida de un programa es un conjunto ordenado (por ejemplo, una tabla, un archivo secuencial, etc.), los casos se deben concentrar en el primero y en el último elemento.

Es recomendable utilizar el ingenio para considerar todos los aspectos y matices, a veces sutiles, en la aplicación del AVL.

Conjetura de errores.

La idea básica de esta técnica consiste en enumerar una lista de equivocaciones que pueden cometer los desarrolladores y de las situaciones propensas a ciertos errores. Después se generan casos de prueba en base a dicha lista (se suelen corresponder con defectos que aparecen comúnmente y no con aspectos funcionales). Esta técnica también se ha denominado generación de casos (o valores) especiales, ya que no se obtienen en base a otros métodos sino mediante la intuición o la experiencia.

No existen directrices eficaces que puedan ayudar a generar este tipo de casos, ya que lo único que se puede hacer es presentar algunos ejemplos típicos que reflejan esta técnica. Algunos valores a tener en cuenta para los casos especiales son los siguientes:

• El valor cero es una situación propensa a error tanto en la salida como en la entrada.

• En situaciones en las que se introduce un número variable de valores (por ejemplo, una lista), conviene centrarse en el caso de no introducir ningún valor y en el de un solo valor. También puede ser interesante una lista que tiene todos los valores iguales.

• Es recomendable imaginar que el programador pudiera haber interpretado algo mal en la especificación.

• También interesa imaginar lo que el usuario puede introducir como entrada a un programa. Se dice que se debe prever toda clase de acciones de un usuario como si fuera “completamente tonto” o, incluso, como si quisiera sabotear el programa.[17]

1.6 Plan de Pruebas.

Un plan de pruebas está constituido por un conjunto de pruebas. Cada prueba debe dejar claro qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad,...), cómo se mide el resultado. Además, un plan de prueba debe especificar en qué consiste la prueba (hasta el último detalle de cómo se

ejecuta), y debe definir cual es el resultado que se espera (identificación, tolerancia,...) ¿Cómo se decide que el resultado es acorde con lo esperado?

Un plan de pruebas contiene información detallada sobre los diferentes casos de prueba (planificados) para su posterior ejecución en el sistema:

- Tipos de pruebas planificadas.
- Entorno operativo de ejecución de las pruebas.
- Elementos auxiliares (p.e., ficheros, tablas, etc.) necesarios para la ejecución de los casos de prueba.
- Responsables de ejecución.

Estas mismas ideas se suelen agrupar diciendo que un caso de prueba debe contener:

- Identificación y objetivo
- Descripción detallada de las entradas
- Modo de ejecución
- Descripción detallada de las salidas esperadas

Y todos y cada uno de esos puntos debe quedar perfectamente documentado. [18]

1.7 Procedimiento de pruebas.

Un procedimiento de prueba especifica como realizar uno o varios casos de prueba o parte de estos. Por ejemplo un procedimiento de prueba puede ser una instrucción para un individuo sobre como ha de realizar un caso de prueba manualmente, o puede ser una especificación de cómo interactuar manualmente con una herramienta de automatización de pruebas para crear componentes ejecutables de pruebas.[20]

Cómo llevar a cabo un caso de prueba puede ser especificado por un procedimiento de prueba pero es a menudo útil reutilizar un procedimiento de prueba para varios casos de prueba y reutilizar varios procedimientos de prueba para varios casos de prueba. Los diseñadores de pruebas intentan crear

procedimientos que puedan ser reutilizables en varios casos de prueba, esto permite usar un conjunto reducido de procedimientos de pruebas con rapidez y precisión para muchos casos de prueba.

Cada caso de prueba precisará varios procedimientos de prueba, quizás uno por cada subsistema de servicio probado en el caso de pruebas. Relacionando de esta forma los procedimientos de prueba con los subsistemas de servicios los procedimientos serian más fáciles de mantener. Cuando se cambia un subsistema de servicios los efectos del cambio que tienen que ver con los procedimientos de prueba pueden entonces estar limitados a los procedimientos de pruebas usados para verificar el subsistema de servicios, y no se vería afectado ningún otro procedimiento de prueba. [20]

1.8 Diseño de Casos de pruebas.

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular ó una función esperada. La entidad más simple siempre es ejecutada como una unidad, desde el comienzo hasta el final, estos casos de pruebas deben verificar:

1. Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos.

2. Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

Formalmente, los casos de prueba escritos contienen principalmente tres partes:

1. Introducción/visión general, contiene información general acerca los casos de prueba.

2. Actividad de los casos de prueba.

3. Resultados. .

En la práctica lo que se prueba puede venir dado por un requisito o una colección de requisitos del sistema cuya implementación justifica una prueba que es posible realizar y que no es demasiado cara de realizar. Los siguientes son casos de pruebas comunes:

1. Un caso de prueba que especifica como probar un caso de uso o un escenario específico de un caso de uso. Un caso de prueba de este tipo incluye una verificación del resultado de la interacción entre los actores y el sistema, que se satisfacen las precondiciones y poscondiciones específicas por el caso de uso y que se sigue la secuencia de acciones especificadas por el caso de uso. Un caso de prueba basado en un caso de uso especifica típicamente una prueba del sistema como “caja negra”, es decir, una prueba del comportamiento observable externamente del sistema.

2. Un caso de prueba que especifica como probar una realización de caso de uso-diseño o un escenario específico de la realización. Un caso de prueba de este tipo puede incluir la verificación de la interacción entre los componentes que implementan dicho caso. Los casos de pruebas basados en una realización de caso de uso típicamente especifican una prueba del sistema como “caja blanca”, es decir, una prueba de la interacción interna entre los componentes del sistema. [19]

1.9 El proceso de pruebas en RUP (Proceso Unificado de desarrollo). Aspectos generales, artefactos, trabajadores y actividades. Metodología.

De forma general se puede decir que RUP es una metodología muy potente de desarrollo de software, que describe como reutilizar los componentes existentes o implementar nuevos componentes con tareas bien definidas, el que nos lleva a obtener un sistema fácil de mantener al que se le puede ir incrementando las posibilidades de reutilización. Asegurando la producción de software de alta calidad con un costo y tiempo predecible para el usuario.

RUP divide el proceso de desarrollo en ciclos, donde se obtiene un producto al final de cada ciclo. Cada ciclo se divide en cuatro Fases: Inicio, Elaboración, Construcción, y Transición. Cada fase concluye con un hito bien definido donde deben tomarse ciertas decisiones.

En varios de los documentos estudiados se plantea la importancia que le da RUP a la ingeniería de pruebas, en aras de lograr la máxima calidad durante el desarrollo del producto en construcción, por ello define claramente aspectos como los objetivos de las pruebas, los principales artefactos y trabajadores, así como las actividades que deben desarrollar los ingenieros de pruebas en las 4 fases del ciclo de vida del proyecto, a todo esto haremos referencia a continuación:

La realización de pruebas persigue varios objetivos que consideramos sean de vital importancia, por lo que se hace necesario:

• 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 sólo al final de la iteración.

• Diseñar y crear las pruebas creando los casos de prueba que especifican qué probar, creando los procedimientos de prueba que especifican 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. [20]

Los principales artefactos definidos en RUP son:

1. Modelo de pruebas: Este modelo describe principalmente cómo se prueban los componentes ejecutables (como las construcciones) en el modelo de implementación con pruebas de integración y de sistema. Describe además cómo deben probarse aspectos específicos del sistema (interfaz de usuario, manual del usuario, etc.). El modelo de pruebas es un conjunto de casos de prueba, procedimientos de prueba y componentes de prueba.

2. Caso de prueba: 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 de probarse. Puede derivarse de un caso de uso del modelo de casos de uso o de una realización de caso de uso del modelo de diseño.

Una **prueba de caja negra**, evalúa el comportamiento externamente observable del sistema, se puede realizar a partir de un caso de prueba que especifica cómo probar un caso de uso del modelo de casos de uso (verificación del resultado de la interacción entre los actores y el sistema, satisfacción de las precondiciones, poscondiciones y secuencias de acciones especificadas por el caso de uso).

Una **prueba de caja blanca**, evalúa la interacción interna entre los componentes del sistema, se puede realizar a partir de un caso de prueba que especifica cómo probar una realización de caso de uso de diseño

o un escenario específico de la realización. Puede incluir la verificación de la interacción entre los componentes que implementan dicho caso de uso.

Pruebas de instalación: Se realizan para verificar que el sistema puede ser instalado en la plataforma del cliente y que funcionará correctamente después de instalado.

Pruebas de configuración: Se realizan para verificar que el sistema funciona bien en diferentes configuraciones, por ejemplo diferentes configuraciones de red.

Pruebas negativas: Intentan provocar que el sistema falle para poder así revelar sus debilidades.

Pruebas de tensión o estrés: Indican problemas con el sistema cuando hay recursos insuficientes o cuando hay competencia por los recursos.

3. Procedimiento de prueba: Específica cómo realizar uno o varios casos de prueba o partes de éstos.

4. Componente de prueba: Automatiza uno o varios procedimientos de prueba o parte de ellos. Pueden ser programados o utilizar una herramienta de automatización de pruebas.

5. Plan de pruebas: Describe la estrategia, recursos y planificación de la prueba. La estrategia de prueba incluye la definición del tipo de prueba a realizar en cada iteración y sus objetivos.

6. Defecto: Es una anomalía del sistema, descubierto en una revisión o un fallo del software.

7. Evaluación de pruebas: Es una evaluación de los resultados de los esfuerzos de prueba.[20]

Los trabajadores definidos en RUP son:

1. Diseñador de pruebas: Planifica las pruebas. Es responsable de la integridad del modelo de pruebas asegurando que el modelo cumple con su propósito. Del modelo de pruebas deben definir y describir los casos de prueba y los procedimientos de prueba. Además son los responsables de la evaluación de las pruebas de integración y de sistema cuando éstas se ejecuten.

2. Ingeniero de componentes: Son responsables de los componentes de pruebas que automatizan algunos de los procedimientos de pruebas.

3. Ingeniero de pruebas de integración: Son los responsables de realizar las pruebas de integración, las cuales se realizan para verificar que los componentes integrados en una construcción funcionan

correctamente juntos, y se derivan a menudo de los casos de pruebas que especifican cómo probar realizaciones de caso de uso de diseño. Debe documentar los defectos en las pruebas de integración.

4. Ingeniero de pruebas de sistema: Son los responsables de realizar las pruebas de sistema sobre los ejecutables. Estas pruebas se llevan a cabo fundamentalmente para verificar las interacciones entre los actores y el sistema. Estas pruebas se derivan a menudo de los casos de pruebas que especifican cómo probar los casos de uso, también se usan para probar el sistema como un todo. Debe documentar los defectos en las pruebas de sistema. [20]

Las principales actividades definidas en RUP se explican a continuación:

1. Planificar pruebas

Al planificar las pruebas, se planifican los esfuerzos de prueba en una iteración y debe llevarse a cabo las siguientes tareas:

- Describir una estrategia de prueba: cuantos flujos básicos y alternativos deben ser probados, cuantas pruebas se realizarán automatizadas y cuántas manuales.
- Estimar los requisitos para el esfuerzo de la prueba: recursos humanos, sistemas necesarios, etc.
- Planificar el esfuerzo de prueba.

Para obtener el plan de prueba, la estrategia de pruebas y los requisitos necesarios de pruebas de una iteración los ingenieros de prueba deben partir de los artefactos de entrada: requisitos adicionales, modelo de casos de uso, modelo de análisis, modelo de diseño, modelo de implementación, descripción de la arquitectura (vistas arquitectónicas de los modelos).

2. Diseñar pruebas

Al diseñar las pruebas deben tenerse los siguientes propósitos:

Identificar y describir los casos de prueba para cada construcción

Para diseñar las pruebas de cada construcción los ingenieros de prueba deben partir de los artefactos de entrada: requisitos adicionales, modelo de casos de uso, modelo de análisis, modelo de diseño, modelo de implementación, descripción de la arquitectura (vistas arquitectónicas de los modelos) y del plan de pruebas

(estrategia y planificación). De esa forma deben obtener los casos de pruebas y los procedimientos de pruebas.

Diseño de casos de pruebas de integración.

Deben ser diseñados los casos de prueba de integración para verificar que los componentes interaccionan entre sí de la forma apropiada después de ser integrados en una construcción. Estos casos de prueba pueden ser derivados de las realizaciones de los casos de uso de diseño (un diagrama de secuencia es parte de la realización del caso de uso diseño) o los diagramas de interacción de las realizaciones de casos de uso, ya que estas realizaciones describen cómo interaccionan las clases y los objetos, y por tanto cómo interaccionan los componentes.

Los diseñadores de pruebas buscan combinaciones de entradas, salida y estado inicial de sistema que den lugar a escenarios interesantes que empleen las clases (y por tanto los componentes) que participan en los diagramas.

Diseño de casos de prueba de regresión.

Algunos de los casos de pruebas de construcciones anteriores pueden ser usados para pruebas de regresión en construcciones subsiguientes, aunque no todos pueden ser utilizados para pruebas de regresión. Los casos de pruebas, para ser usados en pruebas de regresión, deben ser suficientemente flexibles para ser resistentes a cambios, esta flexibilidad debe ser cuidadosa ya que la conversión de un caso de prueba a caso de prueba de regresión supone un esfuerzo de desarrollo extra, luego se debe convertir casos de prueba a casos de prueba de regresión sólo cuando el esfuerzo merezca la pena.

Identificar y estructurar los procedimientos de prueba especificando cómo realizar los casos de prueba.

Se debe reutilizar procedimientos de prueba existentes tanto como sea posible, realizando las modificaciones adecuadas. Los diseñadores de prueba deben crear procedimientos de pruebas que puedan ser reutilizados en varios casos de prueba.

3. Implementar pruebas

El propósito de la implementación de pruebas es automatizar los procedimientos de prueba, creando componentes de pruebas cuando sea posible.

Los componentes de pruebas usan por lo general grandes cantidades de datos de entrada y generan grandes cantidades de datos de salida. Para visualizar estos datos en ocasiones es útil utilizar hojas de cálculo y bases de datos.

4. Realizar pruebas de integración

Esta actividad garantiza que se realicen las pruebas de integración necesarias para cada una de las construcciones creadas en una iteración y se recopilan los resultados de las pruebas. Estas pruebas se llevan a cabo con los siguientes pasos:

1. Realizar las pruebas de integración realizando los procedimientos de pruebas manualmente o ejecutando los componentes de pruebas si existen.

2. Comparar los resultados de las pruebas con los esperados e investigar los resultados de las pruebas que no coinciden con los esperados.

3. Informar de los defectos a los ingenieros de componentes

4. Informar de los defectos a los diseñadores de pruebas, quienes usarán los defectos para evaluar los resultados del esfuerzo de prueba.

5. Realizar prueba de sistema

La prueba de sistema puede empezar cuando las pruebas de integración indican que el sistema satisface los objetivos de calidad de integración fijados en el plan de pruebas de la iteración. Debe realizarse con pasos análogos a las pruebas de integración.

6. Evaluar prueba

El propósito de esta actividad es evaluar los esfuerzos de prueba en cada iteración. Para evaluar los resultados de pruebas estos deben ser comparados con los objetivos esbozados en el plan de pruebas. Los diseñadores de pruebas preparan métricas que les permiten determinar el nivel de calidad de software y qué cantidad de pruebas es necesario hacer. [20]

Luego de analizar el proceso de pruebas que define RUP, se decide utilizar para la ejecución de las pruebas el proceso que describe RUP, puesto que ha sido la metodología de desarrollo de software escogida por el arquitecto del equipo de desarrollo, para aplicar en todo el ciclo de vida del Sistema IPC. Además RUP es una metodología que se puede aplicar en proyectos grandes y pequeños, muy potente para el desarrollo de

software basada en UML (Lenguaje Unificado de Modelado), que describe como reutilizar los componentes existentes o implementar nuevos componentes existentes con tareas bien definidas, llevándonos a obtener un sistema fácil de mantener al que se le puede ir incrementando las posibilidades de reutilización.

Una de las grandes ventajas que tiene RUP son sus casos de uso y sus casos de prueba, ya que los casos de uso facilitan las tareas de programación y los casos de prueba garantizan un plan de pruebas bastante bueno y robusto. Otra ventaja y que constituye una de sus características principales es su enfoque iterativo, implicando con esto que se estará evaluando a lo largo de todo el proyecto, con el fin de encontrar defectos lo antes posible, y poder así reducir el costo por la detección de defectos.

1.10 Conclusiones.

El análisis de las características expuestas sobre el proceso de pruebas nos llevó a la conclusión de que el mismo es de vital importancia a la hora de construir un sistema de software. Ya que define una serie de actividades que garantizan la construcción de un producto final libre de defectos y que satisfaga las necesidades de los clientes.

Se trataron los principales conceptos y definiciones de las pruebas como son: estrategias, niveles, procedimientos, métodos, técnicas, casos de pruebas, etc.

Finalmente la metodología que se usará a lo largo del proceso de pruebas será El Proceso Unificado de Desarrollo (RUP), pues la misma define muy claramente todas las actividades, trabajadores y artefactos que se necesitan para una ingeniería de pruebas eficiente.

CAPÍTULO # 2 DISEÑO Y APLICACIÓN DE PRUEBAS DE SOFTWARE.

2.1 Introducción.

En esencia, en este capítulo se expone el diseño y la aplicación de las pruebas al sistema IPC que se está desarrollando para la ONE, especificando cada uno de los elementos de las pruebas definidas. En el mismo se precisará un Plan de pruebas como base para todo el proceso de pruebas que se llevará a cabo, así como la estrategia de pruebas a seguir y la configuración del entorno de pruebas.

2.2 Cálculo del Índice de Precio al Consumidor: IPC.

El sistema automatizado cuenta con un módulo que contiene toda la información que se necesita para realizar el Cálculo del Índice de Precio al Consumidor mensualmente, el mismo se hace a nivel Nacional, en la Ciudad de la Habana y en Cuba exceptuando la Ciudad de la Habana.

Dicho módulo está compuesto por 7 casos de uso, que recogen los procesos más importantes que se llevan a cabo, considerándose como los más importantes, CU Fusionar ficheros, Cu Emitir Tablas de índice, CU Emitir Tablas de precios, garantizando con estos procesos que se realicen las siguientes funciones:

CU Fusionar ficheros: En este proceso se fusionan todos los ficheros provenientes de provincia, con la información mensual de los mercados, dando paso a que el sistema procese y almacene en la base de datos la información.

Cu Emitir Tablas de índice: En este proceso es donde se lleva a cabo el Cálculo del Índice de Precio a nivel de producto, y posteriormente se realiza un proceso de emisión de una tabla que contiene esta información según el tipo de mercado.

CU Emitir Tablas de precios: En este proceso se realiza el cálculo de los indicadores necesarios para emitir las tablas de precios por mercados, dentro de los indicadores que aquí se calculan está la media aritmética.

Existen en el módulo otra serie de procesos como: Emitir fusión, Autenticar, Gestionar Usuario, Gestionar Clasificador, que serán los procesos adicionales que permitirán que el sistema funcione correctamente y cumpla con todo lo requerido.

2.3 Características a probar.

Las características específicas a probar dentro de cada software pueden variar, en dependencia del software en cuestión y de lo que se pretenda evaluar con la prueba que se va 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.

La aplicación que se está desarrollando será sometida a un proceso de pruebas en el que se verificarán fundamentalmente los atributos de fiabilidad, funcionamiento y usabilidad, para lograr con la comprobación de estos aspectos que el sistema funcione de una manera eficaz. Por otro lado se probarán todos los aspectos funcionales con los que debe contar el sistema para que llegue a la fase final de desarrollo libre de fallas que pueden acortar el tiempo de vida de dicho sistema.

2.4 Plan de Prueba.

La construcción de un buen Plan de Pruebas es la piedra angular, y en consecuencia el principal factor de éxito para la puesta en práctica de un proceso de pruebas que permita entregar un software de mejor nivel. No obstante que cada esfuerzo o proceso de pruebas puede ser diferente y específico, la mayor parte de los proyectos informáticos, sean de nuevos desarrollos o de mantenimiento de aplicaciones, tienen un marco común para la realización de las pruebas. [21]

Dada la importancia que le es conferida al plan de prueba, se ha diseñado un plan que servirá de guía al proceso de pruebas del sistema IPC. Los aspectos generales que lo conforman y los objetivos que se persiguen con el mismo son expuestos a continuación:

En el plan de pruebas definido para el Sistema IPC, se identifican los elementos que serán probados, los recursos necesarios para hacer las pruebas, así como la estrategia de pruebas que se llevará a cabo para lograr un buen diseño de casos de pruebas que permitan encontrar la mayor cantidad posible de defectos al software en cuestión.

Se describen además, las pruebas de unidad que se aplicarán al sistema de software desarrollado, con el objetivo de probar todos los requisitos definidos en la Especificación de Casos de uso y validar y verificar las funcionalidades de la aplicación.

El Plan de Pruebas se precisa el equipo de pruebas, compuesto por dos probadoras, que serán las encargadas de planificar, diseñar y evaluar las pruebas, quienes además realizarán las pruebas de unidad al software en construcción.

Se incluyen las especificaciones de Software y Hardware, teniendo en cuenta las herramientas y los dispositivos que se necesitan para el desarrollo de la aplicación, detallando las versiones utilizadas de cada uno de ellos y los proveedores de los mismos.

Dichas especificaciones son detalladas a continuación:

Especificaciones de software:

La construcción de la aplicación funcionará bajo los principios de la arquitectura cliente-servidor. Por este motivo, el servidor del usuario final debe tener como requerimientos mínimos de software:

- Una computadora personal con sistema operativo Windows 95 o superior.
- SQL Server 2000 como Sistema Gestor de Base de Datos.

Especificaciones de hardware:

Teniendo en cuenta que la arquitectura utilizada será, como se ha mencionado anteriormente, la arquitectura cliente-servidor, para los requerimientos mínimos de hardware, el usuario final debe tener un servidor con las siguientes características:

- Tarjeta de red.
- 128 de RAM o superior.
- 40 GB de HD o superior.
- Pentium II a 1.33 GHz de velocidad de procesador o más.

Un ordenador que sirva de cliente:

- Pentium a 2 GHz de velocidad de procesamiento o superior.
- 128 de RAM o superior.
- Tarjeta de red.

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

Requerimientos funcionales que se deben probar.

R1. Autenticar.
R2.Fusionar datos.
R3.Emitir fusión.
R4. Emitir Tabla de precios.
R5. Emitir Tablas de índices.
R6. Gestionar usuario.
R7. Gestionar clasificador.

En el Plan de pruebas, además de los aspectos a los que ya se ha hecho referencia, se definen otros que se consideran muy importantes como la Estrategia de Pruebas a seguir, la Configuración del Entorno de

Pruebas, los Datos necesarios para hacer las pruebas, los Procedimientos de Pruebas, por último los casos de pruebas.
A todos estos aspectos se hará referencia en los siguientes epígrafes de este capítulo, quedando así conformado todo el Plan de Pruebas.

2.5 Estrategia de Pruebas.

Una estrategia de prueba integra las técnicas de diseño de casos de prueba, en una serie de pasos bien planificados, que llevan a una construcción correcta del software.
Además una estrategia debe incluir pruebas de bajo nivel que verifiquen que todos los pequeños segmentos de código fuente se han implementado correctamente, así como pruebas de alto nivel que validen las principales funciones del sistema frente a los requisitos del cliente. Una estrategia debe proporcionar una guía al profesional y proporcionar un conjunto de hitos para el jefe de proyecto. Debido a que los pasos de la estrategia de prueba se dan a la vez cuando aumenta la presión de los plazos fijados, se debe poder medir el progreso y los problemas deben aparecer lo antes posible.

La estrategia de pruebas desarrollada para el Sistema IPC, tiene los siguientes objetivos:

- Identificar los tipos de prueba a utilizar.
- Definir las técnicas de pruebas que se van a utilizar a lo largo del proceso.
- Definir el entorno en que se van a desarrollar las pruebas.
- Definir los casos de pruebas que se deriven de la aplicación de las técnicas de pruebas.

Las pruebas a definir en una estrategia pueden ser de varios tipos. En el caso del sistema que se está construyendo se ha definido realizar pruebas de Validación y Verificación, con el fin de comprobar todas las funcionalidades del sistema, y lograr la validez de los datos que se entren en cada campo, a continuación se especificará en que consisten la validación y la verificación:

Verificación y validación.

La verificación es: el proceso de evaluación de un sistema o de uno de sus componentes para determinar si los productos de una fase dada satisfacen las condiciones impuestas al comienzo de dicha fase.

La validación es el proceso de evaluación de un sistema o de uno de sus componentes durante o al final del proceso de desarrollo para determinar si satisface los requisitos especificados

De una manera concisa:

- Verificar: significa responder a la pregunta: ¿Estamos construyendo el producto correctamente?, mientras que
- Validar: significa contestar a la pregunta: ¿Estamos construyendo el software correcto?

Mediante este tipo de pruebas se podrá probar la funcionalidad del sistema IPC funcionando como un todo, pues con la verificación se buscará el mayor número posible de discrepancias entre los requerimientos y la ejecución del software y se logra la validación cuando el software funciona de acuerdo con las expectativas razonables del cliente y mediante la misma se pretende demostrar la conformidad de todos los requisitos.

2.5.1 Método de prueba.

El método de prueba que se decidió utilizar en el proceso de pruebas aplicado al sistema IPC es el método de Caja Negra, con el fin de estudiar la especificación de las de las funciones, la entrada y la salida para poder derivar los casos de prueba, definiendo como algo fundamental el probar todas las posibles entradas y salidas del sistema que se está construyendo.

Es decir este método permitirá realizarle pruebas a la interfaz del software y examinar aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del programa, debido se centrará en el estudio de la especificación del software y las funciones que debe realizar, sus entradas y sus salidas.

Con la aplicación de estas pruebas al sistema se podrá realizar un buen diseño para los casos de pruebas cubriendo todas las entradas y salidas que el mismo debe mostrar y haciendo uso de las técnicas que define este método de prueba.

Con la aplicación de estas pruebas al sistema se podrá realizar un buen diseño para los casos de pruebas cubriendo todas las entradas y salidas que el mismo debe mostrar y haciendo uso de las técnicas que define este método de prueba.

2.5.2 Técnicas de pruebas de Caja Negra.

Las técnicas de prueba de Caja Negra escogidas para el Sistema IPC son la de Partición Equivalente y Análisis de los Valores Límites, las mismas ayudarán a diseñar casos de pruebas efectivos, que permiten cubrir el mayor número de clases válidas y no válidas, con las que se garantizará que la entrada y salida de datos al sistema sea la más correcta posible.

La técnica de Partición Equivalente permitirá realizar casos de prueba que cubren el máximo número de entradas, dividiendo estos valores en un número finito de clases de equivalencia que cumplan con la propiedad de que si se prueba un valor representativo de una clase que permite suponer que el resultado obtenido será el mismo que el obtenido probando cualquier otro valor de la clase. Con esta técnica se obtuvieron todos lo valores que el sistema considerará como admisibles y además valores que el sistema debe reconocer como no válidos, dando la posibilidad de encontrar un gran numero de errores al ejecutar la prueba.

El uso de la técnica de análisis de los valores límites permitirá complementar la utilización de la partición equivalente, ya que esta técnica da la posibilidad de explorar más las condiciones límites de un programa ayudando de una manera más eficiente la detección de errores, enmarcándose directamente en los valores que están arriba, abajo y en los márgenes de las clases de equivalencia.

El uso de estas técnicas se evidenciará en el diseño de los casos de pruebas, resaltando con la misma un conjunto de clases válidas y no válidas, que conformarán las pruebas que se ejecuten.

2.6 Configuración del entorno de Prueba.

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

Por ello se tuvo en cuenta a la hora de llevar a cabo todo el proceso de pruebas al Sistema IPC, algunos requerimientos de hardware y de software que hicieron posible un mejor desarrollo de las pruebas, en aras de que las mismas lograran minimizar los errores en la aplicación en desarrollo.
Algunos de los requerimientos que se consideraron necesarios para las pruebas son los referenciados a continuación:

Requerimientos de Software:

- Una PC con Windows 95 o superior.
- Microsoft SQL Server 2000.
- VisualStudio.Net.
- Microsoft Office 2003 ó superior.

Requerimientos de Hardware:

- PC con Microprocesador Pentium III a 2.41 GHz.
- Con 40 GB de Disco Duro.
- 248 MB de memoria RAM.

2.7 Datos de Pruebas.

Cuando se prueban las funciones hay que elegir datos concretos para ejercitar la prueba.
Es imprescindible usar datos "normales", que seguramente serán empleados por un usuario. Pero no vale la pena probar con muchos "datos normales" pues los programas suelen ser monocordes y si funciona con uno, funciona con todos.
Es importante identificar qué rangos de datos pueden alterar el comportamiento del programa y así definir zonas de trabajo. Es imprescindible pasar pruebas con al menos un dato de cada zona, tanto si el programa debe funcionar, como si debe dar un mensaje de error.
La experiencia indica, además, que suelen producirse fallos en los bordes de las zonas, por lo que se recomienda probar siempre con datos extremos. [18]
Para ejecutar los casos de pruebas se creó un juego de datos de pruebas, partiendo de las técnicas de pruebas que se escogieron, donde se almacenan un conjunto de valores de entradas para los diferentes

campos de la aplicación, los cuales fueron escogidos para evaluar si están dentro del rango de posibles valores aceptados de cada campo.

Dicho juego de datos se seleccionó para cada caso de de prueba. Teniendo en cuenta la técnica de prueba escogida, se diseñaron clases de datos admisibles en el caso de probar que la funcionalidad es correcta y datos que no sean admisibles para lograr detectar los posibles errores en cada campo.

Por ejemplo el caso de prueba de Gestionar Clasificador que tiene varias secciones, en la sección de Insertar producto cuenta con varios campos como son el nombre del producto, código, unidad y precio, etc. Dichos campos deben ser verificados, midiendo una serie de criterios que deben cumplir para que la funcionalidad del caso de prueba sea la correcta y se corresponda con los requisitos del cliente. Para ello se diseñaron casos de pruebas que cumplieran las siguientes especificaciones, el nombre del producto solo debe aceptar letras, el precio de un producto solo debe aceptar números y los caracteres punto ó coma, el código del producto no debe exceder los 8 dígitos, entre otras. Los datos necesarios para dichos casos de prueba se muestran en una tabla a continuación:

Tabla # 1: Datos de prueba: "Insertar Producto".

# CP	Nombre Producto	Código prod	Unidad	Precio min	Precio max	Precio min en divisas	Precio max en divisas
1	Arroz	12345678	Libras	3.50	5.60	1.20	2.60
2	Arroz18'[,/	12345678	Libras	3.50	5.60	1.20	2.60
3	Arroz	12345678ais/*	Libras	3.50	5.60	1.20	2.60
4	Arroz	12345678454545	Libras	3.50	5.60	1.20	2.60
5	Arroz	12345678	Libras-=09	3.50	5.60	1.20	2.60
6	Arroz	12345678	Libras	3.50kjf\=	5.60	1.20	2.60
7	Arroz	12345678	Libras	3.50	5.60*-=sham	1.20	2.60
8	Arroz	12345678	Libras	3.50	5.60	1.20jfd$%	2.60
9	Arroz	12345678	Libras	3.50	5.60	1.20	2.60^&*ktog

Los datos de pruebas escogidos para los casos de uso de Fusionar productos, Emitir Fusión, Emitir tablas de precios y Emitir tablas de Índices, son ficheros con los que debe trabajar la aplicación, y que fueron diseñados por el desarrollador del proyecto. Ver ANEXOS [2, 3, 4, 5]

2.8 Procedimientos de pruebas.

Para el Sistema IPC que se está implementando se han definido varios procedimientos de pruebas en dependencia de los diferentes casos de uso que conforman la aplicación, estos procedimientos serán la base para el diseño de casos de pruebas y guiarán la confección de los mismos, puesto que describen como se puede ejecutar un caso de prueba de manera general.

2.8.1 Caso de Uso: <CU Autenticar>

Tabla # 2: Procedimiento: “Autenticar”.

No.	Prueba	Acciones a realizar	Resultado esperado
1	Autenticando usuario.	Seleccione en Administración la opción registrarse. Introduzca el usuario y contraseña. Finalmente introduzca correctamente los datos y presione Entrar.	Se verifica que los datos introducidos son correctos y activa las ventanas del menú correspondiente a este usuario.
1.1	Autenticando usuario.	Introduzca datos que no son válidos.	El sistema muestra el siguiente mensaje:”Acceso Denegado”.

2.8.2 Caso de Uso: <CU Fusionar Datos>

Tabla # 3: Procedimiento: “Fusionar Datos”.

No.	Prueba	Acciones a realizar	Resultado esperado
1	Fusionando Datos.	Seleccione en la opción Fusión y Chequeo, el botón Fusionar productos. Debe escoger el campo de año, se selecciona el que sea correspondiente, al igual que en el mes y se pulsa el botón Examinar, selecciona los ficheros a procesar y pulsa el botón Fusionar.	Se verifica si no se han fusionado los datos seleccionados.
1.1	Fusionando Datos.	Introduzca datos que ya se hayan procesado.	Se verifica que los datos seleccionados no hayan sido fusionados y muestra un mensaje que dice:”Ya se ha procesado con estos datos. ¿Desea rescribir?” <Si> <No>.

2.8.3 Caso de Uso: <CU Emitir fusión >

Tabla # 4: Procedimiento: “Emitir Fusión”.

No.	Prueba	Acciones a realizar	Resultado esperado
1	Emitiendo una fusión.	Seleccione en la opción Fusión y Chequeo, el botón Emitir fusión. Debe escoger Tipo de Mercado, el año y mes que corresponda, después se pulsa el botón Mostrar.	El sistema verifica si se ha procesado con la fecha y el tipo de mercado seleccionado y muestra en pantalla la información correspondiente a los datos entrados.
1.1	Emitiendo una fusión.	Emita una fusión con los datos de la fecha incorrecta.	El sistema verifica si se ha procesado con la fecha y el tipo de mercado seleccionados y muestra en pantalla el siguiente mensaje:”No se ha procesado con la fecha que usted ha seleccionado.”

2.8.4 Caso de Uso: <CU Emitir Tablas de Precios >

Tabla # 5: Procedimiento: “Emitir tablas de precios”.

No.	Prueba	Acciones a realizar	Resultado esperado
1	Emisión de Tablas de precios.	Seleccione el botón Emisión de Tablas de precios. En el campo Región a Emitir se marca la región a la cual se va a emitir, en el campo de Tipo de Mercado se selecciona el que sea correspondiente, al igual que en el año y mes después se pulsa el botón Emitir.	El sistema verifica si se puede emitir con lo datos especificados y muestra por pantalla las tablas de precios.
1.1	Emisión de Tablas de precios.	Emita una tabla de fusión seleccionando datos de manera incorrecta.	El sistema verifica si se puede emitir con lo datos especificados y muestra en pantalla el siguiente mensaje:”No se puede mostrar con dichos datos.”

2.8.5 Caso de Uso: <CU Emitir Tablas de Índices >

Tabla # 6: Procedimiento: “Emitir tablas de índice”.

No.	Prueba	Acciones a realizar	Resultado esperado
1	Emisión de Tablas de Índices.	Seleccione el botón Emisión de Tablas de Índices. En el campo Calcular Región se marca la región a la cual se va a calcular, en el campo de Tipo de Índice se selecciona el que sea correspondiente, al igual que en el año y mes después se pulsa el botón aceptar.	El sistema verifica que los datos sean correctos y muestra por pantalla las tablas con los índices de precios.
1.1	Emisión de Tablas de Índices.	Emita una tabla de Índice seleccionando datos de manera incorrecta.	El sistema verifica que los datos sean correctos muestra por pantalla el siguiente mensaje:”No se puede emitir con dichos campos, inténtelo nuevamente”

2.8.6 Caso de Uso: <CU Gestionar Usuario>

Tabla # 7: Procedimiento: “Gestionar Usuario”.

No.	Prueba	Acciones a realizar	Resultado esperado
1	Creación de usuario.	Pulsando en el botón gestionar usuario Seleccione la sección Crear, dando un clic en el circulo, entonces se activan los campos de dicha sección. Se muestra los componentes para poder introducir un nuevo usuario, el Administrador introduce el nombre de usuario, la contraseña y el tipo de privilegio que va tener. Presiona el botón: <Crear>.	El sistema verifica que el usuario no este creado. En caso de que no exista se crea el usuario.
1.1	Creación usuario.	Cree un usuario, que ya se encuentre registrado.	Se muestra un mensaje de error que este usuario ya existe.
2	Modificando Usuario.	Pulsando en el botón gestionar usuario Seleccione la sección Modificar, dando un clic en el circulo, entonces se activan los campos de dicha sección. Se pulsa en seleccionar usuario se muestra los usuarios creados, y se escoge el usuario al que desea hacerle modificación, se hace modificaciones al usuario, ya sea en el usuario, contraseña o el tipo de privilegio. Presiona botón: <Modificar>.	El sistema verifica los datos y si están correctos actualiza los cambios realizados al usuario.
2.1	Modificando usuarios.	Modifique algún dato y deje campos vacíos.	Se indican la obligatoriedad de los campos.

3	Eliminando usuarios.	Pulsando en el botón gestionar usuario Seleccione la sección Eliminar, dando un clic en el circulo, entonces se activan los campos de dicha sección. Se pulsa en seleccionar usuario se muestra los usuarios creados, y se escoge el usuario al que desea eliminar, Presiona el botón:<Eliminar>	Se muestra el siguiente mensaje por pantalla:" ¿Esta seguro que lo desea eliminar?".<Si> <No> .Presiona <Si>. .
3.1	Eliminando usuarios.	Seleccione la opción de eliminar sin seleccionar ningún usuario.	Se muestra un mensaje de error que no se ha seleccionado ningún usuario

2.8.7 Caso de Uso: <CU Gestionar clasificador >

Tabla # 8: Procedimiento: “Gestionar Clasificador”.

No.	Prueba	Acciones a realizar	Resultado esperado
	Insertando un nuevo producto.	Seleccione en la sección nomencladores, el botón Insertar nuevo producto. Se llenan los campos teniendo en cuenta las siguientes condiciones: para el Nombre Producto acepte solo letras, para el Código prod deben ser números y como máximo deben ser de 8 dígitos, Unidad acepte solo letras, Precio min acepte solo números separados por coma ó puntos, Precio máx, acepte solo números separados por coma ó puntos., Precio min. en divisas, acepte solo números separados por coma ó puntos, Precio máx. en divisas, acepte solo números separados por coma ó puntos. Se pulsa el botón Insertar.	Se verifica que el nombre y el código no existan en otros productos. En caso de que no existan se agrega el producto a la base de datos.
1.1	Insertando un nuevo producto.	Inserte un producto, de manera incorrecta.	Se muestra un mensaje de error señalando los campos incorrectos.
2	Modificando un producto.	Seleccione en la sección nomencladores, el botón modificar producto, se selecciona el código del producto y se activan los campos correspondientes a dicho productos. Se le hacen modificaciones a los campos: código, nombre, unidad, precio min, precio máx. , precio min en divisa, precio máx en divisa, y se pulsa el botón modificar.	El sistema verifica los datos y si están correctos actualiza los cambios realizados al producto.

2.1	Modificando un producto.	Modifique algún dato y deje campos vacíos.	Se indican la obligatoriedad de los campos.
3	Eliminando un producto.	Seleccione en la sección nomencladores, el botón el botón Eliminar Producto. El sistema muestra los productos almacenados. Se selecciona el producto que desea eliminar. Presiona el botón:<Eliminar>.	El sistema muestra el siguiente mensaje por pantalla:" ¿Esta seguro que lo desea eliminar?". <Si> <No>
3.1	Eliminando un producto.	Seleccione la opción de eliminar sin seleccionar ningún producto.	Se muestra un mensaje de error que no se ha seleccionado ningún producto.

2.9 Casos de pruebas.

Un caso de prueba no es más que un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados que se diseñan con el propósito de cumplir un objetivo en particular ó una función esperada.

Los casos de pruebas deben verificar si el producto que se esté realizando satisface los requerimientos del usuario final tal y como se describe en la especificación de los requerimientos y los casos de uso. Además, un caso de prueba debe garantizar que el producto se comporta como se describe en las especificaciones funcionales del diseño.

En fin, el diseño de un buen caso de prueba será el eslabón fundamental del proceso de prueba, pues mediante los mismos se encontrarán la mayor cantidad de defectos posibles y de esta forma se garantizará, que el software en construcción cuando llegue a su fase final este libre de errores.

Los casos de pruebas diseñados para el Sistema IPC se detallan a continuación:

2.9.1 Caso de prueba Autenticar.

Las pruebas diseñadas para este caso de uso son las siguientes:

1. Verificar que los campos de usuario y contraseña sean llenados correctamente.
2. Verificar que no existan campos vacíos a la hora de la autenticación de los usuarios.

CPR 1: <CU Autenticar>

Descripción de la Funcionalidad:

Mediante este caso de uso el sistema da la posibilidad de autenticar a los usuarios del mismo.

Flujo Central:

1. El sistema muestra una interfaz que contiene el usuario y contraseña que se debe introducir.
2. El usuario introduce el nombre y la contraseña.
3. El usuario presiona el botón:<Acceder>.
4. Verifica si el usuario y contraseña entrados son válidos. En caso negativo pasar a 4.1.
5. En caso que sea de tipo 1 (Administrador) se activan todas las opciones del Menú.
6. En caso de que sea de tipo 2(Operador) se activan solo las opciones para ese tipo de usuario.

Flujo Alterno de los eventos.

4.1-El sistema muestra el siguiente mensaje:"Acceso Denegado".

Condiciones de Ejecución:

• Que al menos un usuario se haya autenticado en el sistema.

Tabla # 9: Caso de Prueba: "Autenticar".

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
<Se llenan los campos siguientes: **Usuario:** Roberto **Contraseña:** One123>		<El sistema verifica que los datos introducidos son del tipo 1(Administrador) y activa las ventanas del menú correspondiente a este usuario según sus privilegios>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
<Se llenan los campos siguientes: **Usuario:**		**<**El sistema verifica que los datos introducidos son del tipo 2 (Operador) y	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

JuanCarlos **Contraseña:** Ipc2030>		activa las ventanas del menú correspondiente a este usuario según su privilegio>		
	<Se llenan los campos siguientes: **Usuario:** **Contraseña:** One123>	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el campo usuario está vacío>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<Se llenan los siguientes campos: **Usuario:** JuanCarlos **Contraseña:** >	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el campo contraseña está vacío>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<Se llenan los siguientes campos: **Usuario:** **Contraseña:** >	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que los campos usuario y contraseña están vacíos>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<Se llenan los siguientes	<El sistema verifica la correspondencia	<No ejecutado>	<No se pudo ejecutar la prueba porque la

	campos: **Usuario:** Pepe **Contraseña:** oneadmin>	de los datos y muestra en pantalla un mensaje de error especificando el usuario y la contraseña no se corresponden>		funcionalidad no esta implementada >

Tabla # 10: Registro de defectos y dificultades detectadas. (CP: Autenticar).

Elemento	**No**	**No conformidad**	**Aspecto correspondiente**	**Etapa de detección Código del CP**	**Importancia**	**Recomendación**
<Interfaz >	1	<Cuando se va a la ventana de Administración >	< El nombre de la ventana en la aplicación no coincide con el de la especificación de los casos de uso >	<Implementación y pruebas>	Media	<Verificar que todos los nombres especificados en los casos de uso se correspondan con los de la interfaz de la aplicación, verificar para la segunda iteración>
<Interfaz >	2	<Cuando se va a la ventana de Administració	< El nombre del botón Entrar de la interfaz no coincide con el de la	<Implementación y pruebas>	Media	<Verificar que todos los nombres especificados

		n en la pestaña Registrarse >	especificación de los casos de uso >			en los casos de uso se correspondan con los de la interfaz de la aplicación, verificar para la segunda iteración >
<Interfaz>	3	<Cuando se va a la ventana de Administración en la pestaña Registrarse >	< El nombre del TexBox Usuario de la interfaz no coincide con el de la especificación de los casos de uso >	<Implementación y pruebas>	Media	<Verificar que todos los nombres especificados en los casos de uso se correspondan con los de la interfaz de la aplicación, verificar para la segunda iteración >

2.9.2 Caso de prueba Fusionar Datos.

Las pruebas que se realizan a este caso de uso son para comprobar que los ficheros que llegan de provincia sean fusionados por el sistema de manera correcta, para ello se verifica que:

1. El año y el mes seleccionados sean correctos.
2. Los datos seleccionados no hayan sido fusionados anteriormente.

CPR 2: < CU Fusionar Datos >

Descripción de la Funcionalidad:

Con este caso de uso el sistema permite fusionar los ficheros con las informaciones que llegan de provincia, en la interfaz que correspondiente.

Flujo Central:

1. El operador solicita fusionar ficheros provenientes de provincia.
2. El sistema muestra una interfaz que contiene año y mes a seleccionar.
3. El operador selecciona año y mes.
4. El operador acciona el botón Examinar.
5. Selecciona los ficheros a procesar.
6. Acciona el botón Fusionar.
7. El sistema verifica si no se han fusionado los datos seleccionados. En caso de que Si pasar a 7.1.
8. El sistema procesa y almacena en la base de datos..

Flujo Alterno de los eventos.

7.1 El sistema muestra en pantalla el siguiente cartel:"Ya se ha procesado con estos datos. ¿Desea rescribir?" <Si> <No>

8.1 En caso de que No se termina el caso de uso. En caso de que Si pasar a 8.

Condiciones de Ejecución:

• El operador del sistema esté previamente autenticado.

• Los ficheros deben estar almacenados previamente en algún directorio.

Tabla # 11: Caso de Prueba: "Fusionar Datos".

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
<El operador del sistema selecciona el año, el mes, la extensión de los		< El sistema muestra una ventana de búsqueda para cargar los ficheros a fusionar luego muestra en	<Satisfactorio>	

ficheros y oprime el botón Buscar, se selecciona el fichero deseado y se oprime el botón Fusionar>		pantalla los ficheros correspondientes a la fecha seleccionada y por ultimo muestra los datos de la fusión>		
	<El operador del sistema selecciona el año, el mes y oprime el botón Buscar, se selecciona el fichero deseado y se oprime el botón Fusionar>	<El sistema verifica los datos, no muestra ningún fichero y enseña en pantalla un mensaje de error especificando que el campo de extensión debe ser llenado>	<Insatisfactorio>	< Al presionar el botón Buscar se verifica que no se muestra dicho mensaje de error>
	<El operador del sistema selecciona el año, el mes, la extensión de los ficheros y oprime el botón Buscar, se selecciona el fichero deseado y se oprime el	<El sistema no muestra ningún fichero y enseña en pantalla un mensaje especificando que no hay ficheros con la fecha seleccionada>	<Insatisfactorio>	< Al presionar el botón Buscar se verifica que no se muestra dicho mensaje >

	botón Fusionar>			
	<El operador del sistema selecciona el año, el mes, la extensión de los ficheros y oprime el botón Buscar, se selecciona el fichero deseado y se oprime el botón Fusionar>	<El sistema muestra una ventana de búsqueda para cargar los ficheros a fusionar , luego muestra en pantalla los ficheros correspondientes a la fecha seleccionada y por ultimo y muestra un mensaje que dice:"Ya se ha procesado con estos datos. ¿Desea rescribir?" <Si> <No>>	<Insatisfactorio>	

Tabla # 12: Registro de defectos y dificultades detectados. (CP: Fusionar Datos)

Elemento	**No**	**No conformidad**	**Aspecto correspondiente**	**Etapa de detección Código del CP**	**Importancia**	**Recomendación**
<Interfaz >	1	<Cuando se va a la ventana de Fusión y	<El nombre del caso de uso en el documento de	<Implementación y pruebas>	Media	<Verificar que todos los nombres

		Chequeo en la ventana Fusionar Productos>	especificación de los casos de uso no se corresponde con el nombre de la ventana en la aplicación>			especificados en los casos de uso se correspondan con los de la aplicación, para la segunda iteración>
<Interfaz >	2	<Cuando se va a la ventana de Fusión y Chequeo en la ventana Fusionar Productos>	<No se muestra el mensaje de error, relacionado con la extensión de los ficheros, debido a que la funcionalidad no está implementada >	<Implementación y pruebas>	Alta	<Debe estar disponible para la segunda iteración>
<Interfaz >	3	<Cuando se va a la ventana de Fusión y Chequeo en la ventana Fusionar Productos>	<No se muestra el mensaje de error, especificando que no hay ficheros con la fecha seleccionada, debido a que no está implementada >	<Implementación y pruebas>	Alta	<Debe estar disponible para la segunda iteración>
<Interfaz >	4	<Cuando se va a la ventana de Fusión y Chequeo en	<No se muestra el mensaje de error, especificando que ya se ha fusionado con la fecha	<Implementación y pruebas>	Alta	<Debe estar disponible para la segunda iteración>

		la ventana Fusionar Productos>	seleccionada, debido a que no está implementada >			
Especificación de casos de uso.	5	<En el flujo central del caso de uso >	<Faltan muchos mensajes Informativos en la especificación de los casos de uso del sistema, para guiar al usuario en el uso del sistema.>	<Implementación y pruebas>	Alta	< Debe estar disponible para la segunda iteración>

2.9.3 Caso de prueba Emitir Fusión.

A este caso uso se realizaran las pruebas siguientes:

1. Verificar que los campos del año, mes y el tipo de mercado seleccionados sean correctos.
2. Verificar que se haya fusionado algún dato anteriormente para poder mostrarlo.

CPR 3: <CU Emitir Fusión>

Descripción de la Funcionalidad:

Mediante este caso de uso el sistema muestra los datos de una fusión ya realizada a través de la interfaz correspondiente con el caso de uso.

Flujo Central:

1. El operador solicita Mostrar Fusión.
2. El sistema muestra una interfaz que contiene tipo de mercado, año y mes a seleccionar.
3. El operador selecciona tipo de mercado y fecha.
4. El operador oprime el botón: <Mostrar>.
5. El sistema verifica si ha procesado con la fecha y mercado seleccionados. En caso de que no pasar a 5.1.

6. El sistema muestra en pantalla la información correspondiente a los datos entrados por el actor.

Flujo Alterno de los eventos

5.1. Se muestra por pantalla el siguiente mensaje:"No se ha procesado con la fecha que usted ha seleccionado."

Condiciones de Ejecución:

• El operador del sistema este autenticado previamente.

• En el sistema debe haber datos de alguna fusión realizada.

Tabla # 13: Caso de Prueba: "Emitir Fusión".

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
< El usuario selecciona el año, mes, el tipo de mercado y oprime el botón Mostrar>		< El sistema verifica si se ha procesado con los datos seleccionados y muestra en pantalla la información correspondiente a los datos entrados por el actor>	<Satisfactorio>	
	< El usuario selecciona el año, mes, el tipo de mercado y oprime el botón Mostrar>	< El sistema verifica si se ha procesado con los datos seleccionados y muestra en pantalla el siguiente mensaje:"No se ha procesado con la fecha que usted ha seleccionado.">	<Insatisfactorio>	< Al presionar el botón Mostrar se verifica que no se muestra dicho mensaje de error>

Tabla # 14: Registro de defectos y dificultades detectados. (CP: Emitir fusión)

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección Código del CP	Importancia	Recomendación
Interfaz	1	< Cuando se va a la ventana de Fusión y Chequeo en la pestaña de Emitir Fusión >	<El nombre de la ventana de la interfaz no coincide con el de la especificación de los casos de uso>	<Implementación y pruebas>	Media	<Verificar que todos los nombres especificados en los casos de uso se correspondan con los de la interfaz de la aplicación, para la segunda iteración>
Interfaz	2	< Cuando se va a la ventana de Fusión y Chequeo en la pestaña de Emitir Fusión >	<No se muestra el mensaje de error, que especifica que no se ha procesado con los datos seleccionados, debido a que la funcionalidad no está implementada >	<Implementación y pruebas>	Alta	<Debe estar disponible para la segunda iteración>
Especificación de casos de uso.	3	<En el flujo central del caso de uso>	<Faltan muchos mensajes Informativos en la especificación de	<Implementación y pruebas>	Alta	<Debe estar disponible para la segunda iteración>

			los casos de uso del sistema, para guiar al usuario en el uso del sistema.>			

2.9.4 Caso de prueba Emitir Tablas de Precios.

Las pruebas que se realizaran a este caso de uso son las siguientes:

1. Verificar que los campos año, mes, región a emitir y el tipo de mercado seleccionados sean correctos.
2. Verificar que los precios solicitados para la emisión estén registrados en la base de datos.

CPR 4: <CU Emitir tablas de precios>

Descripción de la Funcionalidad:

Mediante este caso de uso el sistema da la posibilidad de ver las tablas de los indicadores de precio en la interfaz correspondiente.

Flujo Central:

1. El operador solicita emisión de tablas de precios.
2. El sistema pide que especifique el tipo de mercado, año y mes a procesar.
3. El operador especifica el tipo de mercado, año y mes.
4. El operador del sistema oprime el botón: <Emitir>.
5. El sistema verifica si puede emitir con lo datos especificados. En caso de que no pasar a 5.1.
6. Muestra por pantalla las tablas de precios.

.

Flujo alterno de los eventos

5.1-Se muestra en pantalla el siguiente mensaje:”No se puede mostrar con dichos datos.”

Condiciones de Ejecución:

• El operador debe estar previamente autenticado.

• Los precios que el sistema necesita para la emisión de las tablas deben estar almacenados previamente.

Tabla # 15: Caso de Prueba: “Emitir tablas de precio”.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
< El operador selecciona el año, el mes, el tipo de mercado, y oprime el botón Emitir >		**<** El sistema verifica si se puede emitir con lo datos especificados y muestra por pantalla las tablas de precios.>	<Satisfactorio>	
	< El operador selecciona el año, el mes, el tipo de mercado, y oprime el botón Emitir >	<El sistema verifica si se puede emitir con lo datos especificados y muestra en pantalla el siguiente mensaje:”No se puede mostrar con dichos datos.”>	<Insatisfactorio>	< Al presionar el botón Mostrar se verifica que no se muestra dicho mensaje de error>

Tabla # 16: Registro de defectos y dificultades detectados. (CP: Emitir tablas de precio)

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección Código del CP	Importancia	Recomendación
Interfaz	1	<Cuando se va a la ventana de Emitir tablas de Precio >	<No se muestra el mensaje de error, especificando que no se ha procesado con los datos seleccionados>	<Implementación y pruebas>	Alta	<Debe estar disponible para la segunda iteración>
Especific	2	<En el flujo	<Faltan muchos	<Implementac	Alta	<Debe estar

ación de casos de uso.		central del caso de uso >	mensajes Informativos en la especificación de los casos de uso del sistema, para guiar al usuario en el uso del sistema.>	ión y pruebas>		disponible para la segunda iteración>

2.9.5 Caso de prueba Emitir Tablas de Índice.

Las pruebas realizadas a este caso de uso son:

1. Verificar que los campos año, mes, región a calcular y el tipo de índice seleccionados sean correctos.
2. Verificar que existan precios con los datos seleccionados.

CPR 5: < CU Emitir Tablas de índices.>

Descripción de la Funcionalidad:

Mediante este caso de uso el sistema da la posibilidad de ver las tablas con los índices de precios a través de su interfaz correspondiente.

Flujo Central:

1. El operador solicita emisión de tablas de índices.
2. El sistema le pide que especifique el tipo de mercado, año y mes a procesar.
3. El operador especifica el tipo de mercado, año y mes.
4. El operador oprime el botón: <Aceptar>.
5. El sistema verifica si puede emitir con lo datos especificados. En caso de que no pasar a 5.1.
6. El sistema muestra por pantalla las tablas de índices.

Flujo alterno de los eventos

5.1- Se muestra en pantalla el siguiente mensaje:”No se puede emitir con dichos campos, inténtelo nuevamente”.

Condiciones de Ejecución:

• Que el operador esté previamente autenticado.

• Los índices de precio que el sistema necesita para mostrar las tablas con los precios estén almacenados previamente.

Tabla # 17: Caso de Prueba: “Emitir tablas de índice”.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
<El usuario selecciona el año, el mes y el tipo de mercado >		< El sistema verifica que los datos sean correctos y muestra por pantalla las tablas con los índices de precios >	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<El usuario selecciona el año, el mes y el tipo de mercado >	<El sistema verifica que los datos sean correctos muestra por pantalla el siguiente mensaje de error:”No se puede emitir con dichos campos, inténtelo nuevamente”>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

Tabla # 18: Registro de defectos y dificultades detectados. (CP: Emitir tablas de índice)

Elemento	**No**	**No conformidad**	**Aspecto correspondiente**	**Etapa de detección Código del CP**	**Importancia**	**Recomendación**
Interfaz	1	<Cuando se va a la ventana de	< El nombre de la ventana en la	<Implementación y	Media	<Verificar que todos los

		Cálculo de los Índices de Precio >	aplicación no coincide con el de la especificación de los casos de uso >	pruebas>		nombres especificados en los casos de uso se correspondan con los de la interfaz de la aplicación, para la segunda iteración>
Interfaz	2	<Cuando se va a la ventana de Cálculo de los índices de precio >	< El nombre del botón Mostrar de la interfaz no coincide con el de la especificación de los casos de uso >	<Implementación y pruebas>	Media	<Verificar que todos los nombres especificados en los casos de uso se correspondan con los de la interfaz de la aplicación, para la segunda iteración >

2.9.6 Caso de prueba Gestionar Usuario.

Este caso de uso cuenta con las siguientes pruebas:

1. Verificar que los datos para crear un usuario sean correctos.
2. Verificar que los datos para modificar un usuario sean correctos.
3. Verificar que los datos para eliminar un usuario sean correctos.

CPR 6: <Gestionar Usuario>

Descripción de la Funcionalidad:

Mediante este caso de uso el sistema gestiona los usuarios que se autentican en el mismo y les da los permisos correspondientes a cada uno de ellos.

Flujo Central:

1. Solicita la gestión de usuarios.
2. El sistema muestra una interfaz que contiene cuatro opciones:
    a. Crear un nuevo usuario.
    b. Modificar usuario.
    c. Eliminar usuario.

El administrador selecciona una opción. Si decide la opción a) pasar a la sección Crear. Si decide la opción b) pasar a la sección Modificar. Si decide la opción c) pasar a la sección Eliminar.

Flujo alterno de los eventos:

Sección Crear: Línea 3

En caso de que exista el usuario se muestra el siguiente mensaje:"El usuario ya existe". Regresa a la línea 1 de la sección.

Sección Eliminar: Línea 3

En caso de oprima la opción No regresa a la línea 1.

Condiciones de Ejecución:

- El administrador del sistema este previamente autenticado.
- Debe haber usuarios insertados para poder modificarlos o eliminarlos.
- Deben estar llenos todos los campos necesarios para poder insertar un nuevo usuario.
- Los nombres de los usuarios no deben repetirse, ni las contraseñas.

Sección Crear

1. El sistema muestra los componentes para poder introducir un nuevo usuario.

2. El Administrador introduce el nombre de usuario, la contraseña y el tipo de privilegio que va tener. Presiona el botón: <Crear>.

3. Verifica que el usuario no esté creado. En caso de que no exista se crea el usuario.

Tabla # 19: Caso de Prueba: “Gestionar usuario, Sección Crear”.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
<Se llenan los campos siguientes: **Usuario:** Juan Carlos **Contraseña:** Juan2190 **Confírmela:** Juan2190 **Privilegio:** Administrador y se oprime el botón Crear >		<El sistema verifica que el usuario no haya sido creado y lo crea >	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
<Se llenan los campos siguientes: **Usuario:** Alberto **Contraseña:** Albert91 **Confírmela:**		< El sistema verifica que el usuario no haya sido creado y lo crea >	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

Albert91 **Privilegio:** Operador y se oprime el botón Crear >				
	<Se llenan los campos siguientes: **Usuario:** **Contraseña:** Juan2190 **Confírmela:** Juan2190 **Privilegio:** Administrador y se oprime el botón Crear >	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el campo usuario esta vacío>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<Se llenan los campos siguientes: **Usuario:** Juan Carlos **Contraseña:** **Confírmela:** Juan2190 **Privilegio:** Administrador	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el campo contraseña esta vacío>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

	y se oprime el botón Crear >			
	<Se llenan los campos siguientes: **Usuario:** Juan Carlos **Contraseña:** Juan2190 **Confírmela: Privilegio:** Administrador y se oprime el botón Crear >	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el campo confírmela esta vacío>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	Se llenan los campos siguientes: **Usuario:** Juan Carlos **Contraseña:** Juan2190 **Confírmela:** Juan2190	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que no se ha seleccionado ningún privilegio>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

	Privilegio: y se oprime el botón Crear >			
	Se llenan los campos siguientes: **Usuario:** Juan Carlos **Contraseña:** Juan2190 **Confírmela:** Juan2190 **Privilegio:** operador y se oprime el botón Crear >	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el nombre de usuario no coincide con el privilegio seleccionado>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

	Se llenan los campos siguientes: **Usuario:** Alberto **Contraseña:** Albert91 **Confírmela:** alber **Privilegio:** operador y se oprime el botón Crear >	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando la contraseña introducida no coincide con la confirmada>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

Sección Modificar

1. El sistema muestra los usuarios creados.
2. El administrador selecciona el usuario al que desea hacerle modificación.
3. El administrador hace modificaciones al usuario, ya sea el nombre de usuario, contraseña o privilegio. Presiona botón: <Modificar>
4. El sistema actualiza los cambios realizados.

Tabla # 20: Caso de Prueba: “Gestionar usuario, Sección Modificar”.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
< El administrador selecciona el usuario,		<El sistema verifica los datos y modifica los campos>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

modifica los campos que desee y oprime el botón Modificar >				
< El administrador selecciona el usuario a modificar, modifica un campo especifico y oprime el botón Modificar >		<El sistema verifica los datos y modifica el campo>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<El administrador selecciona el usuario y los campos a modificar y oprime el botón Modificar>	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que no hay correspondencia entre la contraseña nueva introducida y la confirmada>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<El administrador selecciona el usuario a	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

	modificar y llena los campos con datos incorrectos, oprime el botón Modificar>	que los datos introducidos son incorrectos>		
	< El administrador selecciona el usuario, modifica el campo usuario y oprime el botón Modificar >	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que ya existe un usuario con ese nombre>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada>
	< El administrador selecciona el usuario, modifica el campo contraseña y oprime el botón Modificar >	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que la contraseña introducida ha sido utilizada anteriormente>		

Sección Eliminar

1. El sistema muestra los usuarios creados.
2. El administrador selecciona el usuario que desea eliminar. Presiona el botón:<Eliminar>.
3. Se muestra el siguiente mensaje en pantalla:" ¿Está seguro que lo desea eliminar?".

<Si> <No>

3. Presiona <Si>.

Tabla # 21: Caso de Prueba: “Gestionar usuario, Sección Eliminar”.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
<El administrador selecciona el usuario y oprime el botón Eliminar>		<El sistema verifica los datos y muestra en pantalla el siguiente mensaje de confirmación: “Esta seguro que lo desea eliminar? <Si> <No> ” >	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<El administrador no selecciona ningún usuario de la lista y oprime el botón eliminar>	<El sistema muestra en pantalla un mensaje de error especificando que no se ha seleccionado ningún usuario>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

2.9.7 Caso de prueba Gestionar Clasificador.

Las pruebas que se realizarán a este caso de uso cubren los siguientes aspectos:

1. Verificar que los datos introducidos en los campos sean los correctos.
2. Verificar que el campo nombre del producto acepte solo letras.
3. Verificar que el campo código del producto acepte solo 8 dígitos.
4. Verificar que el campo código del producto acepte solo números.
5. Verificar que el campo Unidad acepte solo letras.

6. Verificar que el campo precio min. acepte solo números separados por coma ó puntos.
7. Verificar que el campo precio máx. acepte solo números separados por coma ó puntos.
8. Verificar que el campo precio min. en divisas acepte solo números separados por coma ó puntos.
9. Verificar que el campo precio máx. en divisas acepte solo números separados por coma ó puntos.
10. Verificar que no existan campos vacíos a la hora de insertar un producto.
11. Verificar que existan productos para poder eliminarlos.

CPR 7: <CU Gestionar Clasificador>

Descripción de la Funcionalidad:

Mediante este caso de uso el sistema permite insertar o eliminar un producto en la aplicación a través de dicha interfaz.

Flujo Central:

1. El administrador del sistema solicita gestionar clasificador.
2. El sistema muestra una interfaz que contiene dos opciones:

a) Insertar producto.

b) Eliminar producto.

3. El administrador selecciona una opción. Si decide la opción a) pasar a la sección Insertar. Si decide la opción b) pasar a la sección Eliminar.

Flujo alterno de los eventos

Sección Insertar: Línea 3

En caso de que exista el código y el nombre en otro producto se muestra el siguiente mensaje:"El producto ya se encuentra almacenado".Regresa a la línea 1 de la sección.

Sección Eliminar: Línea 3

En caso No regresa a la línea 1.

Sección Modificar: Línea 3

En caso no regresa a la línea 1.

Condiciones de Ejecución:

- El administrador del sistema este previamente autenticado.
- Debe haber productos insertados para poder modificarlos o eliminarlos.

- Deben estar llenos todos los campos necesarios para poder insertar un nuevo producto.
- Deben estar llenos todos los campos necesarios para poder modificar un nuevo producto.
- No deben existir productos con el mismo nombre, ni el mismo código.

Sección Insertar. Flujo central.

1. El sistema muestra los componentes para poder introducir un nuevo producto.
2. El Administrador introduce el código, nombre, precio máximo, precio mínimo y unidad de medida del producto. Presiona el botón: <Insertar>
3. Verifica que el nombre y el código no existan en otros productos. En caso de que no existan se agrega el producto a la base de datos.

Tabla # 22: Caso de Prueba: “Gestionar clasificador, Sección insertar”.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
<Se llenan los campos siguientes: **Nombre Producto:** Arroz **Código prod:** 12345678 **Unidad:** Libras **Precio min**.: 3.50 **Precio máx**.: 5.60 **Precio min. en divisas:** 1.20 **Precio máx.**		**<** El sistema verifica que los datos del nombre y el código no existan en otros productos, en caso de que no existan el sistema inserta el producto>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

en divisas: 2.60>				
	< Se llenan los campos siguientes: **Nombre Producto:** Arroz18'[,/ **Código prod:** 12345678 **Unidad:** Libras **Precio min.:** 3.50 **Precio máx.:** 5.60 **Precio min. en divisa**s: 1.20 **Precio máx. en divisas:** 2.60	<El sistema verifica los datos y muestra un mensaje de error especificando que el nombre del producto solo acepta letras >	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<Se llenan los campos siguientes: **Nombre Producto:** Arroz **Código prod:** 12345678ais/* **Unidad:** Libras	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el código del producto solo acepta números>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

	Precio min.: 3.50 **Precio máx**.: 5.60 **Precio min. en divisas:** 1.20 **Precio máx. en divisas:** 2.60>			
	<Se llenan los campos siguientes: **Nombre Producto:** Arroz **Código prod:** 12345678454545 **Unidad:** Libras **Precio min**.: 3.50 **Precio máx**.: 5.60 **Precio min. en divisas:** 1.20 **Precio máx. en divisas:** 2.60>	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el número máximo de dígitos que acepta el campo código del producto es 8 dígitos>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<Se llenan los campos siguientes: **Nombre Producto:** Arroz **Código prod:**	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el campo de la unidad	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

	12345678 **Unidad:** Libras-=09 **Precio min**.: 3.50 **Precio máx**.: 5.60 **Precio min. en divisas:** 1.20 **Precio máx. en divisas:** 2.60>	solo se letras>		
	<Se llenan los campos siguientes: **Nombre Producto:** Arroz **Código prod:** 12345678 **Unidad:** Libras **Precio min**.: 3.50kjf\= **Precio máx**.: 5.60 **Precio min. en divisas:** 1.20 **Precio máx. en divisas:** 2.60>	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el campo precio min. solo acepta números y los caracteres punto y coma >	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<Se llenan los campos	<El sistema verifica los datos y muestra	<No ejecutado>	<No se pudo ejecutar la prueba porque la

	siguientes: **Nombre Producto:** Arroz **Código prod:** 12345678 **Unidad:** Libras **Precio min**.: 3.50 **Precio máx**.: 5.60*-=sham **Precio min. en divisas:** 1.20 **Precio máx. en divisas:** 2.60>	en pantalla un mensaje de error especificando que el campo precio máx. solo acepta números y los caracteres punto y coma>		funcionalidad no esta implementada >
	<Se llenan los campos siguientes: **Nombre Producto:** Arroz **Código prod:** 12345678 **Unidad:** Libras **Precio min**.: 3.50 **Precio máx**.: 5.60 **Precio min. en divisas:** 1.20jfd$% **Precio máx. en divisas:**	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el campo de precio min. en divisas solo acepta números y los caracteres punto y coma>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

	2.60>			
	<Se llenan los campos siguientes: **Nombre Producto:** Arroz **Código prod:** 12345678 **Unidad:** Libras **Precio min**.: 3.50 **Precio máx**.: 5.60 **Precio min. en divisas:** 1.20 **Precio máx. en divisas:** 2.60^&*ktog >	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que el campo de precio máx. en divisas solo acepta números y los caracteres punto y coma>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<Se llenan los campos siguientes: **Nombre Producto:** **Código prod:** 12345678 **Unidad:** Libras **Precio min**.: 3.50 **Precio máx**.:	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que todos los campos deben estar llenos >	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

	Precio min. en divisas: 1.20 **Precio máx. en divisas:** 2.60 >			

Sección Modificar. Flujo central.

1-El sistema muestra los productos almacenados.

2-El administrador selecciona el producto y los componentes que desea modificar del mismo. Presiona el botón:<Modificar>

3-Se muestra el siguiente mensaje por pantalla:" ¿Esta seguro que desea realizar las modificaciones?".

<Si> <No>

4-Presiona <Si>

5-El sistema modifica el producto.

Tabla # 23: Caso de Prueba: "Gestionar usuario, Sección Modificar".

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
<El administrador selecciona el producto a modificar y llena los campos que desea cambiar, oprime el botón Modificar y confirma la		**<**El sistema verifica que los datos seleccionados sean correctos, muestra en pantalla un mensaje indicando: ¿Esta seguro que desea realizar las modificaciones?". <Si> <No>, y modifica los campos >	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

modificación >				
El administrador selecciona el producto a modificar y va a modificar un campo en específico.		<El sistema verifica que los datos estén correctos y muestra un mensaje indicando: ¿Esta seguro que desea realizar las modificaciones?". <Si> <No> y modifica los campos >	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<El administrador selecciona el producto a modificar y llena los campos con datos incorrectos oprime el botón Modificar>	<El sistema verifica los datos y muestra en pantalla un mensaje de error especificando que los datos introducidos son incorrectos>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

Sección Eliminar. Flujo central.

1. El sistema muestra los productos almacenados.
2. El administrador selecciona el producto que desea eliminar. Presiona el botón:<Eliminar>.

3. Se muestra el siguiente mensaje por pantalla:" ¿Esta seguro que lo desea eliminar?". <Si> <No>

Tabla # 24: Caso de Prueba: "Gestionar usuario, Sección Eliminar".

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
<El administrador selecciona el producto a eliminar y oprime el botón Eliminar>		<El sistema verifica los datos y muestra en pantalla el siguiente mensaje de confirmación: "Esta seguro que lo desea eliminar? <Si> <No> " >	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >
	<El administrador no selecciona ningún producto de la lista y oprime el botón eliminar>	<El sistema muestra en pantalla un mensaje de error especificando que no se ha seleccionado ningún producto>	<No ejecutado>	<No se pudo ejecutar la prueba porque la funcionalidad no esta implementada >

Tabla # 25: Registro de defectos y dificultades detectados. (CP: Gestionar Clasificador).

Elemento	**No**	**No conformidad**	**Aspecto correspondiente**	**Etapa de detección Código del CP**	**Importancia**	**Recomendación**
Interfaz	1	< Cuando se va a la ventana de	<El nombre de la ventana de la interfaz	<Implementación y	Media	<Verificar que todos los

		nomencladores >	no coincide con el de la especificación de los casos de uso>	pruebas>		nombres especificados en los casos de uso se correspondan con los de la interfaz de la aplicación>
Interfaz	2	< Cuando se va a la ventana de nomencladores >	<Falta una interfaz que describa el flujo central del caso de uso gestionar clasificador, pues aparece directo las interfaces de insertar y eliminar producto>	<Implementación y pruebas>	Media	<Verificar que todos los nombres especificados en los casos de uso se correspondan con los de la interfaz de la aplicación>
Interfaz	3	< Cuando se va a la ventana de nomencladores >	< La interfaz del caso de uso gestionar clasificador no coincide con la documentada en la especificación del caso uso, cuenta con una interfaz de Mostrar Canasta >	<Implementación y pruebas>	Media	<Verificar que todos los nombres especificados en los casos de uso se correspondan con los de la interfaz de la aplicación>

2.10 Conclusiones.

En este capítulo quedaron explicadas las características que tendrá el plan de pruebas diseñado para el sistema IPC en construcción, así como los datos con los que se realizaran las pruebas, la estrategia a seguir en el diseño de los casos de pruebas y la configuración del entorno que se empleará en todo la etapa de pruebas.

También se obtuvieron todos los procedimientos de pruebas, los cuales permitieron una mejor comprensión del funcionamiento del sistema, pudiendo utilizar los mismos para el diseño y ejecución de los casos de pruebas.

CAPÍTULO # 3 ANÁLISIS DE LOS RESULTADOS PARCIALES.

3.1 Introducción.

Este capítulo hará referencia al análisis de los resultados obtenidos luego de diseñar y aplicar el proceso de pruebas al Sistema IPC, Para poner en práctica dicho análisis se hará una pequeña valoración de la cantidad de clases de equivalencia diseñas y las no conformidades encontradas en cada una de las pruebas aplicadas, por ultimo se creará una clasificación de los errores mayormente encontrados.

3.2 Análisis de los resultados obtenidos en las pruebas de Caja Negra.

Las pruebas fueron diseñadas y realizadas por 2 probadoras del equipo de desarrollo. Los resultados de los errores comprenden los resultados de los casos diseñados según las entradas, los errores de interfaz y de las pruebas de requerimientos.

Por cada caso de uso se realizó un caso de prueba compuesto por las clases de equivalencias diseñadas, algunas con datos válidos y otras con datos no válidos, obteniéndose un resultado en cada una de las pruebas que se ejecutaron.

La siguiente tabla resume la información derivada del diseño y aplicación de las pruebas, para los 7 casos de pruebas diseñados se obtuvo un total de 46 clases válidas y no válidas. Solo se pudieron ejecutar 3 casos de pruebas con 8 clases de válidas y no válidas debido a que la aplicación se encuentra aún en la fase de implementación y muchas de las funcionalidades necesarias para ejecutar las pruebas no están disponibles.

Tabla # 26: “Resultado del diseño y aplicación de pruebas”

Nombre del Caso de prueba	Clases de equivalencia diseñadas
Autenticar	En este caso de prueba se diseñaron 6 Clases de equivalencias, 2 de ellas fueron válidas y 4 inválidas.
Fusionar Datos	En este caso de prueba se diseñaron 4 Clases de equivalencias, 1 de ellas fue válida y 3 inválidas.
Emitir Fusión	En este caso de prueba se diseñaron 2 Clases de equivalencias, 1 de ellas fue válida y la otra inválida.
Emitir Tablas de Índices	En este caso de prueba se diseñaron 2 Clases de equivalencias, 1 de ellas fue válida y la otra inválida.
Emitir Tablas de Precio	En este caso de prueba se diseñaron 2 Clases de equivalencias, 1 de ellas fue válida y la otra inválida.
Gestionar Clasificador	En este caso de prueba se diseñaron 15 Clases de equivalencias, 4 de ellas fueron válidas y 11 inválidas.
Gestionar Usuario	En este caso de prueba se diseñaron 16 Clases de equivalencias, 5 de ellas fue válida y 11 inválidas.

3.3 Clasificación de los errores.

Para poder cuantificar la cantidad de errores encontrados se confeccionó una clasificación, estructurada de la siguiente forma:

- ✓ Errores graves todos aquellos que impiden el cumplimiento de una funcionalidad que debe generar el sistema y no está especificada.
- ✓ Errores leves todos aquellos relacionados con los nombres de los campos y las ventanas de la aplicación que no se corresponden con la especificación de casos de uso del sistema.

Los 3 casos de pruebas que se pudieron ejecutar contaron con 8 clases de equivalencia probadas, encontrándose 8 errores de tipo graves y 2 de tipo leves con un total de 10 errores. En los casos de prueba que no se pudieron ejecutar también se encontraron 8 errores de tipo leve. A continuación se relacionarán los errores más repetidos por casos de usos.

Tabla # 27: “Principales errores detectados”.

Nombre del CU	Errores	Tipo de Errores	Recomendaciones
Fusionar Datos	Los errores mas repetidos son: El nombre del caso de uso en el documento de especificación de los casos de uso no se corresponde con el nombre de la ventana en la aplicación. No se muestra el mensaje de error, especificando que ya se ha fusionado con la fecha seleccionada, debido a que no está implementada la funcionalidad que define esto. Faltan muchos mensajes Informativos en la	Los errores que se pueden encontrar son graves (4), la minoría de tipo leve (1).	Se recomienda que el programador se ajuste a la descripción de los casos de uso del analista a la hora de programar el sistema, para una mejor comprensión del mismo y que el usuario no se pierda a la hora de utilizar la aplicación. Sería bueno utilizar más mensajes para guiar al cliente, ya que el flujo central de los eventos no queda

	especificación de los casos de uso del sistema, para guiar al usuario en el uso del sistema.		muy claro. Se recomienda programar las funcionalidades que tienen que ver con algunos mensajes de error que no aparecen en la aplicación cuando se produce uno de ellos, para que el cliente sepa a donde debe dirigirse para corregir dicho error, pues aunque la aplicación no ejecuta la acción se debe especificar.
Emitir Fusión	Los errores mas repetidos son: El nombre de la ventana de la interfaz no coincide con el de la especificación de los casos de uso. No se muestra el mensaje de error, que especifica que no se han procesado los datos seleccionados	Los errores que se pueden encontrar son graves (2), la minoría del tipo leve (1).	Se recomienda que el programador se ajuste a la descripción de los casos de uso del analista a la hora de programar el sistema, para una mejor comprensión del mismo y que el usuario no se pierda a la hora de utilizar la aplicación. Se recomienda

			programar las funcionalidades que tienen que ver con algunos mensajes de error que no aparecen en la aplicación cuando se produce uno de ellos, para que el cliente sepa a donde debe dirigirse para corregir dicho error, pues aunque la aplicación no ejecuta la acción se debe especificar.
Emitir Tablas de Precio	Los errores mas repetidos son: No se muestra el mensaje de error, especificando que no se ha procesado con los datos seleccionados.	Los errores que se pueden encontrar son graves (2).	Se recomienda programar las funcionalidades que tienen que ver con algunos mensajes de error que no aparecen en la aplicación cuando se produce uno de ellos, para que el cliente sepa a donde debe dirigirse para corregir dicho error, pues aunque la aplicación no ejecuta la acción se debe

			especificar.
Autenticar	Los errores mas repetidos son: El nombre de la ventana de la interfaz no coincide con el de la especificación de los casos de uso. No se encontraron errores graves, pues no se pudo ejecutar la prueba porque la funcionalidad no estaba implementada	Los errores que se pueden encontrar son leves (3).	Se recomienda que se verifique que todos los nombres especificados en los casos de uso se correspondan con los de la interfaz de la aplicación y que para la culminación del sistema se apliquen las pruebas diseñadas.
Emitir Tablas de índices	Los errores mas repetidos son: El nombre de la ventana de la interfaz no coincide con el de la especificación de los casos de uso. No se encontraron errores graves, pues no se pudo ejecutar la prueba porque la funcionalidad no estaba implementada	Los errores que se pueden encontrar son leves (2).	Se recomienda que se verifique que todos los nombres especificados en los casos de uso se correspondan con los de la interfaz de la aplicación y que para la culminación del sistema se apliquen las pruebas diseñadas.
Gestionar Clasificador	Los errores mas repetidos son: El nombre de la ventana de la interfaz no coincide con el	Los errores que se pueden encontrar son leves (3).	Se recomienda que se verifique que todos los nombres especificados en los

	de la especificación de los casos de uso. No se encontraron errores graves, pues no se pudo ejecutar la prueba porque la funcionalidad no estaba implementada.		casos de uso se correspondan con los de la interfaz de la aplicación y que para la culminación del sistema se apliquen las pruebas diseñadas.
Gestionar Usuario	En este caso de prueba no se encontraron errores ya que no se pudo ejecutar la prueba porque la funcionalidad no estaba implementada y tampoco se detectaron errores de tipo leve.		Se recomienda que para la culminación del sistema se le apliquen estas pruebas ya diseñadas.

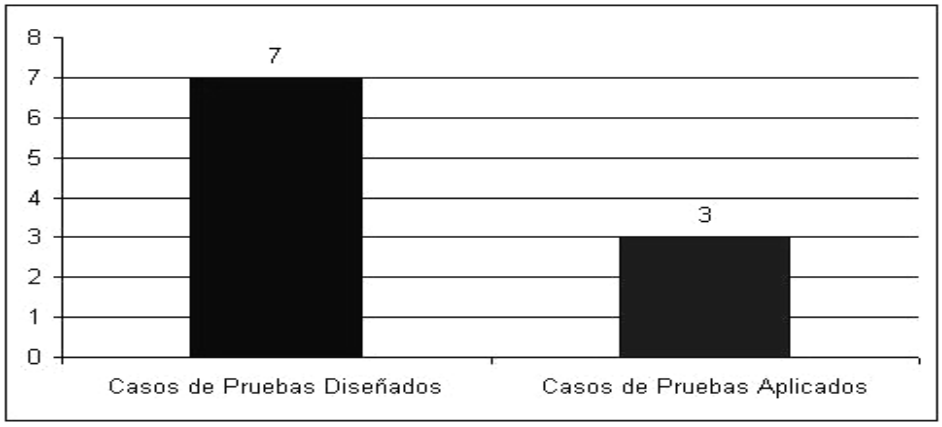
3.4 Comparación entre los resultados diseñados y los obtenidos.

La comparación de los resultados obtenidos de manera general durante el diseño de las pruebas y la fase de aplicación de los mismas muestra que la aplicación se encuentra en una fase de implementación que debe avanzar progresivamente, pues muchas de las funcionalidades que se debían probar aún no están disponibles, impidiendo la ejecución de todas las pruebas y por ende la detección de una mayor cantidad de errores.

Para poder analizar todos los resultados obtenidos de manera general se han confeccionado una serie de gráficos donde se demuestran los resultados del diseño de las pruebas y los resultados de la aplicación de las pruebas.

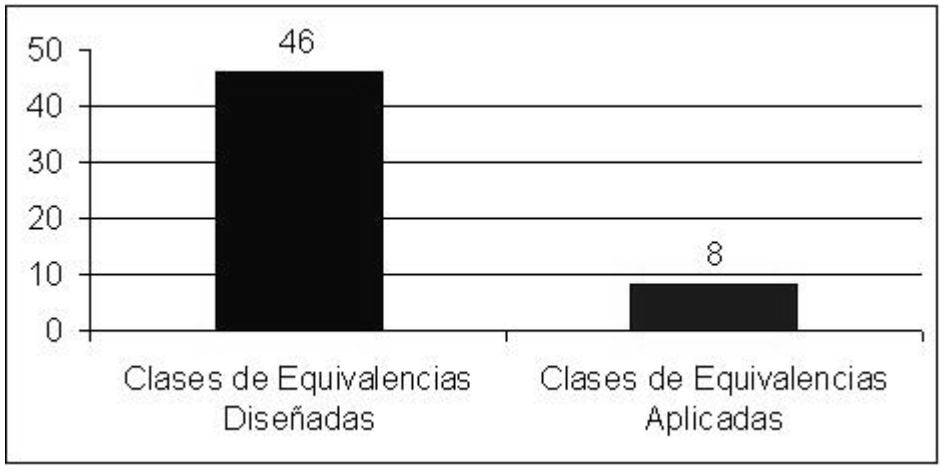
La gráfica #1 representa el total de casos de pruebas diseñados diferenciando los que se le pudieron ejecutar las pruebas. Se tiene que para un total de 7 casos de pruebas solo se pudieron ejecutar las pruebas a 3 casos de pruebas, (Estos fueron Fusionar Datos, Emitir Fusión, Emitir tablas de precios).

Gráfica # 1: “Total de Casos de prueba”



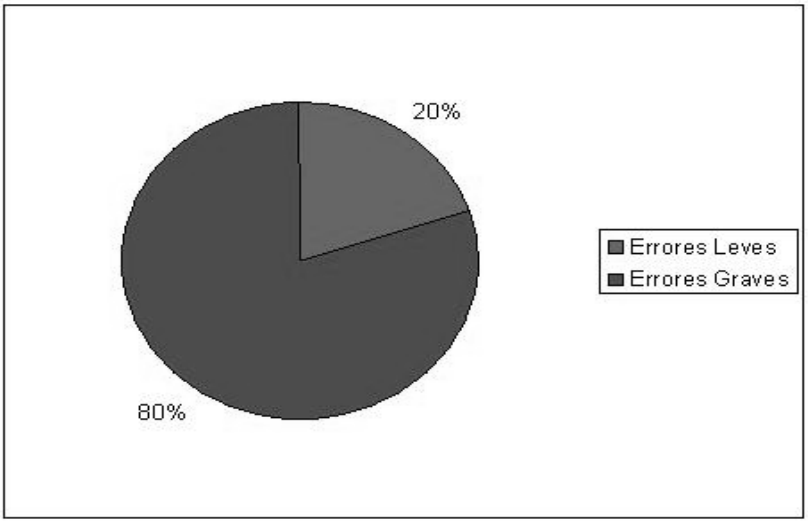
La gráfica #2 muestra el total de clases de equivalencia diseñadas (46), para los 7 casos de pruebas y de estas las que se pudieron aplicar (8), en los 3 casos de pruebas que fueron ejecutados.

Gráfica # 2: “Total de Clases de Equivalencia”.



La gráfica #3 expone los resultados de los errores encontrados en los casos de prueba a los que se le aplicaron las pruebas diseñadas.

Gráfica # 3: “Errores en las pruebas aplicadas”

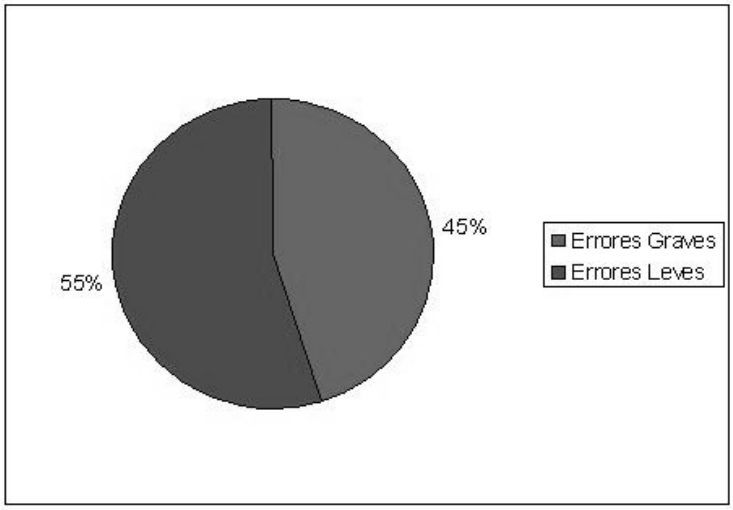


Los 3 casos de pruebas que se pudieron ejecutar contaron con 8 clases de equivalencia probadas, encontrándose 8 errores de tipo graves y 2 de tipo leves con un total de 10 errores, lo que representa un 80 % de errores graves y un 20 % de errores leves, del total de errores encontrados.

La cuarta gráfica resume el total de los errores encontrados debido a que en los casos de pruebas que no se pudieron ejecutar las mismas también se encontraron que no afectan la funcionalidad del sistema pero deben de ser corregidos.

En estos casos de prueba se encontraron 8 errores de tipo leve que sumados a los errores ya encontrados hacen un total de 18 errores, (10) de tipo leves que representan un 55 % del total y 8 de tipo grave que representan 45 % del total como se muestra en la gráfica #4.

Gráfica # 4: “Total de errores”.



3.5 Conclusiones.

Con este capítulo concluye el proceso de pruebas diseñado y aplicado al Sistema IPC, el desarrollo del mismo permitió obtener el resultado de las pruebas que se aplicaron mostrando la efectividad de las pruebas pues se pudieron encontrar errores de tipos graves y leves, pudiendo minimizar de esta forma la cantidad de errores de en las funcionalidades probadas.

CONCLUSIONES

La aplicación de las pruebas al sistema garantizó una mayor fiabilidad y utilidad del mismo, además permitió que todas las funcionalidades que se especificaron al inicio por parte de los clientes fueran cumplidas satisfactoriamente, contribuyendo así al logro de la calidad del producto final en el tiempo requerido y con el mínimo de los costos.

Se analizó toda la documentación acerca de la ingeniería de pruebas, llegando a comprender todas las definiciones, conceptos y aspectos en general que conforman este tema. Se comprendió la utilidad de las técnicas de pruebas de Caja Negra en el diseño de casos de pruebas certeros, haciendo posible que se cubrieran la mayor cantidad de casos de pruebas posibles. Se resaltó además la necesidad de utilizar el Proceso Unificado de Desarrollo RUP, por ser una metodología que cuenta con un proceso de pruebas como flujo de trabajo muy importante con actividades, artefactos y los trabajadores claramente definidos. Además por ser un proceso guiado por casos de uso, centrado en la arquitectura e iterativo e incremental.

Se confeccionó un Plan de pruebas que incluye todo lo necesario para la planificación del proceso de pruebas, donde se definió una estrategia de pruebas a seguir con el fin de guiar la etapa de pruebas y se estableció la configuración del entorno en el cual iban a ser ejecutadas definiendo todos los recursos de software y hardware necesarios en este.

Se diseñaron todos los casos de prueba posibles elaborándose un juego de datos para su validación apoyándose en los procedimientos de prueba construidos.

Se elaboró un juego de datos que formaron parte de las clases válidas y no válidas de los casos de pruebas. Se construyeron los procedimientos en vistas de poder utilizarlos en los casos de pruebas. Se diseñaron todos los casos de pruebas posibles, haciendo uso de las técnicas de pruebas definidas en la estrategia y del juego de datos, así como de los procedimientos. Se realizó una evaluación de los resultados derivados de los casos pruebas.

RECOMENDACIONES

Luego de aplicar todo el proceso de pruebas de caja negra al Sistema IPC y demostrar la importancia que tiene en el desarrollo de software, para lograr una adecuada detección de errores, y valorar el beneficio que aportan al software en construcción las fallas encontradas, se recomienda:

- ✓ Dar continuidad al proceso de pruebas aplicado al Sistema IPC.
- ✓ Comenzar la segunda revisión del Ingeniero de pruebas aplicando las pruebas que han sido diseñadas en la primera revisión y quedaron por ejecutarse.
- ✓ Realizarle otros tipos de pruebas al sistema, como son las pruebas de Caja Blanca.

REFERENCIAS BIBLIOGRÁFICAS

1. Hernández., M.A.C. *Modelo para Pruebas de Software y auditoría en Entorno Microsoft.Net.* **Volume**, 50
2. IEEE, *Standard Glossary of Software Engineering Terminology.* Software Engineering Standars Collection, 1990.
3. Pressman, R., *Ingenieria de Software. Un enfoque practico*. Segunda ed. 1998, España: European Edition. McGraw Hill.
4. Cig_Labs. *The Home of Groundbreaking Software Quality*. 2002 [cited; Available from: http://:www.cigitallabs.com/resources/definitions/software_testing.html.
5. Rice, R.W., *Surviving the top 10 challenges of software test automation*. 2002: Cross Talk: The journal of Defense Software Engineering.
6. Hetzel, B., *The complete guide to software testing*. Segunda ed. 1998: QED Information Science.
7. Peña, J.M.F., *Pruebas de integración para componentes de software*. 2002, Mexico.
8. D'Onofrio, D.L. *Prueba de Software*. 2002 [cited; Available from: http://www.elguille.info/Clipper/probando.htm.
9. Mañas, J.A. *Pruebas de Programas*. 1994 [cited; Available from: http://www.upu.es/potel/user/jotrofer/pascal/testing.htm.
10. Davis, A., *Principios del Desarrollo de Software*. 1995: McGraw.
11. Myers, G., *The art of software testing*. 1979: John Wiley.
12. Pressman, R., *Ingeniería de Software. Un enfoque Práctico*. Quinta ed. 2002, España: European Edition. McGraw Hill.
13. Kaner C, F.J., H.Q Nguyen, *Testing Computer Software*. Segunda ed. 1993: Van Nostrang Reinold.
14. Moreno, G.A. *Pruebas*. 2003 [cited; Available from: http://lsi.urg.es/~arroyo/inndoc/doc/pruebas.
15. Ferrón María José and González Juan Pablo. *Sistemas de Programas* 2001 [cited; Available from: http://www.ldc.usb.ve/~teruel/ci3711/test1/index.html#3%20Pruebas%20de%20m□dos.
16. Alfonso, M.I. *Tema 8. Pruebas del software.* 2001 [cited.
17. *Diseño de la Interfaz de Usuario. Pruebas del Software.* [cited.

18. Mañas, J.A. *Calidad de las Pruebas*. 2003 [cited; Available from: http://www.lab.dit.upm.es/~lprg/index.html.

19. *Caso de prueba*. Enero 2007 [cited; Available from: http://es.wikipedia.org/wiki/Caso_de_prueba.

20. Ivar Jacobson, G.B., James Rumbaugh, *El proceso unificado de desarrollo de software.* . 2000, Madrid.

21. Gónzalez, C. *Un plan de Pruebas Exitoso*. 2002 [cited; Available from: http://www.americaxxi.cl/modules.php?name=News&file=article&sid=20.

GLOSARIO DE TÉRMINOS

IPC: Índices de Precio al Consumidor. (Es una medida estadística de la evolución de los precios de los productos y servicios que consumen los hogares cubanos.)

ONE: Oficina Nacional de Estadísticas. (Entidad encargada de archivar y procesar gran cantidad de información de muchas esferas de nuestra sociedad.)

Peso: Porcentaje en importancia que representa un IPC específico con respecto al General.

RUP: Proceso Unificado de Desarrollo.

UML: Lenguaje Unificado de Modelado.

CP: Caso de prueba.