Universidad de las Ciencias Informáticas

Facultad 3

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

Título: "Diseño y desarrollo de casos de pruebas para el Sistema de Gestión de Inventario y Almacenes (SIGIA)"

Autora: Nayrobi Fuentes Telles

Tutora: Ing: Yenisleidy Rendón Vigil

Consultante: Dra. Neida Aragón

Junio, 2007

DECLARACIÓN DE AUTORÍA

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

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

Nayrobi Fuentes Telles

_____

Yenisleidy Rendón Vigil

_____

AGRADECIMIENTOS

A mi mamá por toda la dedicación y atención que tuvo en mis estudios.

A mi papá por apoyarme siempre.

A mis hermanos Noharis y Lázaro Jesús por darme todo el amor del mundo.

A nana por ser mi segunda madre y por todo su amor y dedicación a mi formación.

A mis tías Aidita, Marian, Irma, por todo su amor y ayuda en mis estudios.

A mi novio Adrian por darme fuerzas y amor para continuar.

A Berenice , Leamnet, Yenier y a Roxana por brindarme su ayuda en los momentos más difíciles de mi vida y por su amistad tan sincera.

A mi amiga Ana del Rosario y a Pascual por ayudarme y brindarme tanto cariño.

A mis amigos por permitirme compartir con ellos todos estos años de estudios.

A mis primos por quererme tanto.

A toda mi familia.

Dedico mi trabajo de diploma

“A mi Madre Adela Telles, que aunque ya no está a mi lado todo lo que soy y seré en la vida se lo agradezco a ella porque en cada tropiezo, ante cada momento difícil de mi vida siempre tuve de ella un gesto de amor y dedicación. Siempre me apoyó para que estudiara y me superara y me enseñó que con esfuerzo todo se puede alcanzar. Gracias por haber sido mi motor impulsor ante todos mis años de estudios.”

El presente trabajo, titulado “Diseño y desarrollo de casos de pruebas para el Sistema de Gestión de Inventario y Almacenes (SIGIA)” se realizó con el propósito de diseñar y ejecutar los casos de prueba para el sistema de Gestión de Inventarios y Almacenes. Esto permite el control de la calidad del proceso de desarrollo, y crea las bases para el aseguramiento de la calidad del mismo.

Para ello fue necesario realizar una fundamentación teórica en la que se abordan los diferentes conceptos emitidos por varios autores referentes a aspectos relacionados con calidad, calidad de software, pruebas de software, entre otros, que sustentan el hilo conductor del marco teórico referencial; por otra parte diseñó un Plan de Pruebas que recoge la metodología a seguir en el diseño y aplicación de los casos de pruebas a los módulos Nomencladores y Administración del SIGIA. Se utilizaron las técnicas de caja negra y caja blanca para la realización de las pruebas, lo que permitió la detección de los errores de los casos de uso y de la lógica del programa de la aplicación SIGIA.

PALABRAS CLAVE

Pruebas de Software, Calidad de Software

TABLA DE CONTENIDOS

INTRODUCCION

El rol que desempeña la tecnología en el mundo de hoy es de suma importancia para el hombre y la sociedad. En un período de tiempo bastante breve con relación a otros momentos de avances científicos en la historia, el hombre ha aprendido a utilizar la tecnología en su beneficio en una amplia gama de actividades, tanto cotidianas como netamente científicas, industriales o comerciales.

Uno de los usos más relevantes que se le ha dado a los avances tecnológicos ha sido apoyar y beneficiar la producción de software, ya que esta industria no ha acabado de salir de la fase artesanal producto de la desorganización y de la falta de planificación, pero hoy se dedican esfuerzos a mejorar lo que se hizo mal en el pasado.

En vista de las exigencias de la industria del software de hoy en día, ha sido necesario y de suma importancia hacer software de una buena calidad para poder entrar en el mercado. La Universidad de la Ciencias Informáticas juega un papel de gran importancia, pues su objetivo principal es producir software para Cuba y para el resto del mundo. Para el cumplimiento de este objetivo la universidad tiene proyectos productivos de diferentes temas; y para logar un nivel óptimo de calidad en cada software producido se hace necesaria la realización de un plan de pruebas y de una estrategia de pruebas que asegure el cumplimiento de los requisitos de calidad.

El incremento de la complejidad de los sistemas de software aumenta a su vez la necesidad de asegurar su calidad. La fase de prueba del sistema ayuda a asegurar la calidad del software. La mayoría de las pruebas en la industria se desarrolla a nivel del sistema; sin embargo, muchas técnicas de prueba del sistema suelen realizarse al final del proceso de desarrollo, por lo que estas pruebas suelen realizarse de manera superficial e incompleta.

Las pruebas de software son una técnica clásica para la verificación y validación del software, junto a otras alternativas de evaluación como revisiones, auditorías, métricas. Las pruebas de software no son más que la ejecución de un programa con el fin de detectar errores. Para ello, el diseño de las pruebas se basa en la creación de casos de prueba cuya ejecución permitirá

observar posibles síntomas de defectos. Toda la capacidad de detección de defectos de las pruebas se basa en la consideración de cualquier tipo de discrepancia entre la salida obtenida y la salida esperada (según lo indicado en las especificaciones) como síntoma de un problema en el software. En consecuencia, todas las técnicas de diseño de casos de prueba existentes se basan en tomar las especificaciones como referencia de comportamiento de la aplicación.

Desde la década del 70, el tema de las pruebas de software ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software, los cuales han realizado gran cantidad de investigaciones al respecto. El 50 por ciento del tiempo empleado en el proyecto y más del 50 por ciento del costo total, era utilizado en las pruebas o en desarrollo del sistema [Myers, 2004].

La inexperiencia existente acerca de la realización de pruebas de software es un problema que afecta a la UCI, ya que el producto debe ser probado y evaluado para garantizar que cumpla con los requisitos de los clientes. En los proyectos no se realizan las pruebas desde las primeras fases de desarrollo del software, no se lleva una guía, ni un archivo de los defectos que puede tener el software durante la fase de construcción.

Durante los primeros años de este siglo el país se ha centrado en un proceso de informatización de la sociedad y dentro de este proceso, la Universidad de la Ciencias Informáticas ha jugado un papel rector, en la misma se han estado desarrollando un conjunto de proyectos con el objetivo de informatizar los procesos que se llevan a cabo en esta.

Un proyecto que se está desarrollando es un sistema para la gestión de los inventarios de los almacenes, válido para cualquiera institución o empresa. En función de alcanzar en el sistema para la gestión de inventarios en almacenes un producto con calidad, es necesaria la realización de pruebas a este. Estas deben registrar todos los defectos relacionados con los requerimientos del producto sobre la base de los requisitos. De no lograrse este aspecto se crea un conflicto entre el producto a lograr y la satisfacción del cliente.

En el proyecto “Sistema de Gestión de Inventario y Almacenes” que se desarrolla en la Universidad de las Ciencias Informáticas no se aplica un plan de pruebas que logre evaluar la

calidad final del producto a pesar de que ya existe una metodología que regula este proceso para que los niveles de calidad del mismo se corresponda con los requisitos del cliente.

Y de esto se desprende el siguiente **problema científico a resolver:** ¿Cómo aplicar las pruebas de software al producto Sistema de Gestión de Inventario para que los niveles de calidad del mismo se corresponda con los requisitos del cliente?

Para dar solución al problema científico se planteó como **objetivo general:** Diseñar y desarrollar casos de pruebas para el proyecto Sistema de Gestión de Inventario de Almacenes para garantizar que los niveles de calidad del mismo se corresponda con los requisitos del cliente.

Objeto de estudio: Calidad del software.

Campo de acción: Pruebas de software sobre el sistema de Gestión de Inventario de Almacenes.

Con el objetivo de guiar la investigación se plantea la siguiente **Hipótesis:** Si se realiza adecuadamente el diseño y desarrollo de los casos de pruebas para el proyecto Sistema de Gestión de Inventario a Almacenes, se logrará un producto final con la calidad requerida para cumplir con los requisitos del cliente.

Los **objetivos específicos** trazados para la investigación son:

- Estudiar el estado del arte de las estrategias de pruebas a utilizar en proyectos productivos; y los principales conceptos y herramientas relacionados con la calidad de software.
- Realizar un estudio de los casos de uso del software Sistema de Gestión de Inventario.
- Diseñar casos de pruebas y probarlos en cada uno de los casos de uso estudiados del sistema de Gestión Inventario.
- Realizar un análisis de los resultados obtenidos en las pruebas realizadas a cada uno de los casos de uso del Sistema de Gestión de Inventario, y así poder arribar a las conclusiones y recomendaciones de la investigación para el caso en cuestión.

Con el objetivo de darle solución a los objetivos trazados durante la investigación, se utilizaron varios métodos de investigación: entre los métodos teóricos se aplicaron el histórico-lógico, el analítico-sintético, el hipotético-deductivo y el analítico-sistémico; y entre los métodos empíricos, la experimentación.

***Aportes prácticos esperados en el trabajo*.**

Con el desarrollo de este trabajo se espera obtener una estrategia de prueba y un plan de pruebas que organice la estructura de la realización de las pruebas al software del Sistema de Gestión de Inventario y Almacenes.

Un sistema o software que cumpla con las normas de calidad requeridas para así garantizar la validación de los requisitos propuestos por el cliente para el Sistema de Gestión de Inventario y Almacenes, hacer una validación de los algoritmos que me permitan descubrir los errores de los mismos. Establecer una secuencia de pasos para la realización de los métodos de Caja Negra y Caja Blanca.

Breve referencia de cada capítulo

La tesis esta estructurada en dos capítulos:

En el capítulo 1 se expone una reseña bibliográfica de los principales conceptos de Gestión de Inventario, Software de Gestión, Calidad, Calidad de Software, Sistemas de Calidad del Software, Diseño de Pruebas y por último se plantea la necesidad del diseño de prueba para el proyecto Sistema de Gestión de Inventario.

En el capítulo 2 se realiza la construcción de los diferentes artefactos a desarrollar en la tesis. Se expone la estrategia de prueba a seguir para el desarrollo de sistema SIGIA, el modelo de prueba, los casos de prueba que se aplicarán, la configuración del entorno de prueba, se construye el plan de prueba, así como los datos de prueba y los logs de prueba.

En un epígrafe se realiza un análisis de los resultados obtenidos al aplicar las pruebas a los casos de uso del sistema.

CAPITULO 1 ESTUDIO DEL ESTADO DEL ARTE DEL DESARROLLO DE LAS PRUEBAS DEL SOFTWARE Y SU IMPORTANCIA.

1.1 Introducción

El análisis bibliográfico permite profundizar en los aspectos investigativos, así como conocer los distintos criterios y valoraciones que tienen diferentes autores sobre una temática determinada. Este análisis resulta de gran importancia, pues constituye una base sólida que fundamenta los métodos, procedimientos, técnicas y conceptos manejados en la investigación.

El marco teórico de la investigación (Figura 1) que se desarrolla en este capítulo está dirigido a recoger los aspectos más significativos donde se aborden las principales temáticas relacionadas con el desarrollo y el análisis de la elaboración de casos de pruebas, con el objetivo de lograr una mejor calidad en el Sistema de Gestión de Inventario y Almacenes, y los principales conceptos y herramientas relacionados con la gestión de inventario y los sistemas de gestión de la calidad de software, que serán aplicados para resolver las problemáticas que confronta el proyecto Sistema de Gestión de Inventario y Almacenes (SIGIA).

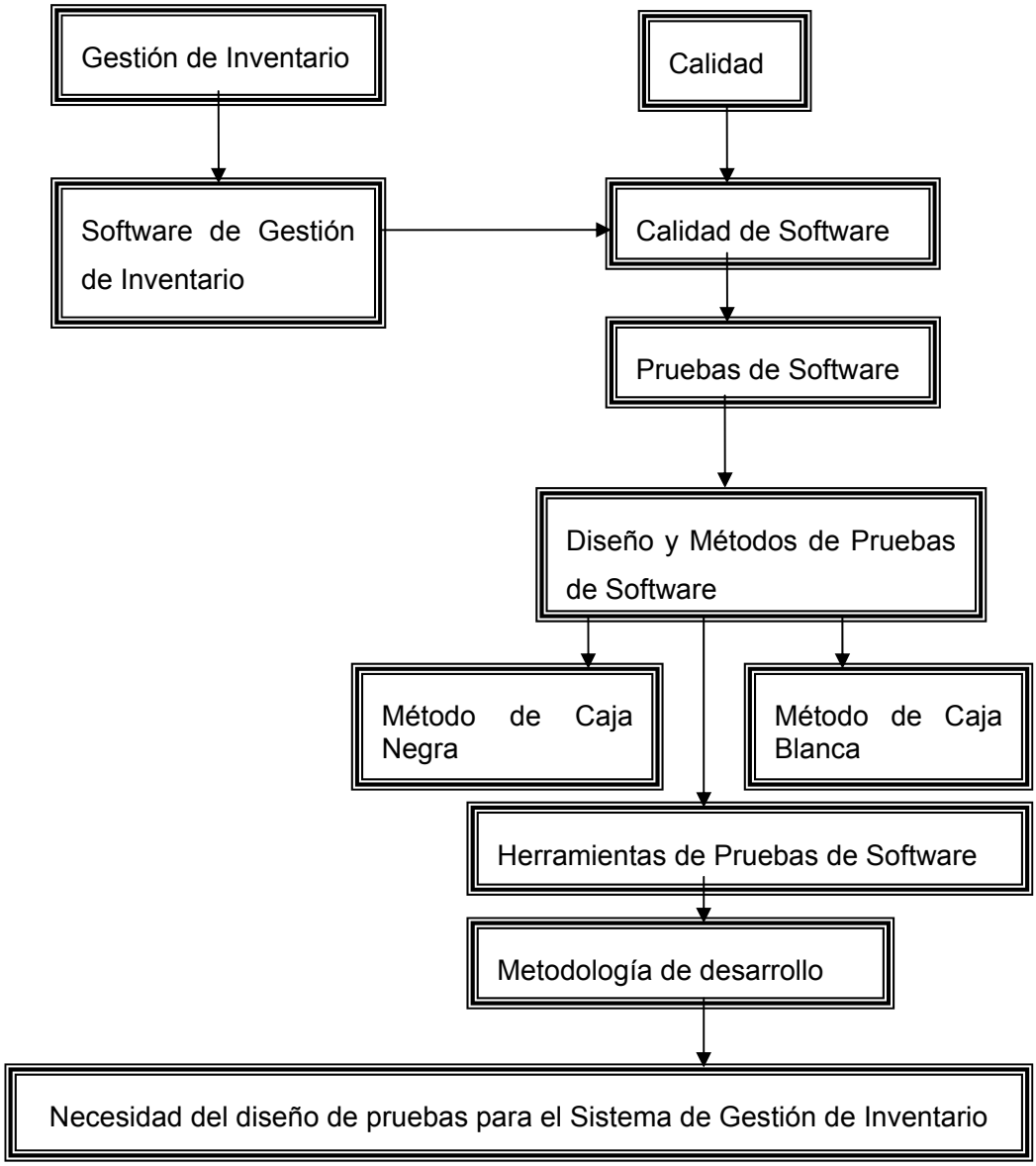


Figura 1. Hilo Conductor de la Fundamentación Teórica

1.2 Definición y características de la Gestión de Inventario

Para comenzar, es necesario definir algunos conceptos clave sobre los que se basa la gestión de inventario de los almacenes de una empresa.

Se entiende por Gestión de Inventarios, "todo lo relativo al control y manejo de las existencias de determinados bienes, en la cual se aplican métodos y estrategias que pueden hacer rentable y productivo la tenencia de estos bienes y a la vez sirve para evaluar los procedimientos de entradas y salidas de dichos productos" [JPulido, 2006].

La Gestión de Inventarios parte de tres actividades básicas.

Determinación de las existencias: Se refiere a todos los procesos necesarios para consolidar la información de las existencias físicas de los productos a controlar, y estos procesos se detallan como:

- Toma física de inventarios
- Auditoria de existencias
- Evaluación a los procedimientos de recepción y ventas (entradas y salidas)
- Conteos cíclicos

Análisis de inventarios: Es un análisis estadístico que se realiza para verificar si las existencias que fueron previamente determinadas son las que se deberían tener en una empresa. Para lograr este control, existen metodologías aplicables como:

- Formula de Wilson (máximos y mínimos)
- Just in Time (justo a tiempo)

Control de producción: Es la evaluación de todos los procesos de manufactura realizados en el departamento a controlar. Se verifica la transformación de materia prima en productos terminados para su comercialización, los métodos más utilizados para lograr este fin son:

- MRP (planeación de recursos de manufactura)
- MPS (plan maestro de producción)

Los inventarios o stocks son materiales y suministros que una empresa o institución posee, ya sea para vender o para abastecer al proceso productivo. [Cuatrecasas - 2003]

Los inventarios de acuerdo a las características físicas de los objetos a contar, pueden ser de los siguientes tipos:

Inventarios de materia prima o insumos: Son aquellos en los cuales se contabilizan todos aquellos materiales que no han sido modificados por el proceso productivo de las empresas.

Inventarios de materia semi-elaborada o productos en proceso: Son materiales que han sido modificados por el proceso productivo de la empresa, pero que todavía no son aptos para la venta.

Inventarios de productos terminados: Se contabilizan todos los productos que van a ser ofrecidos a los clientes.

La gestión de inventarios implica dos costos básicos:

Costos de penalización por inexistencia de los materiales: básicamente falta de productos a ofrecer en el mercado.

Costos de almacenamiento: representan costos tanto en capital inmovilizado como en costos de gestión física y administrativa de estos inventarios. Se considera que la gestión de inventario es importante para el desarrollo de cualquier empresa y para el aseguramiento de la calidad de sus productos, o de los servicios que presta, ya que los inventarios representan una gran cantidad de los bienes de cada institución. Por la importancia del tema se ha hecho necesario la automatización de los procesos de la gestión de inventario, por lo que surgen diferentes software de gestión de inventario para el aprovechamiento de cada recurso de la empresa.

1.3 Software de Gestión de Inventario

Un ejemplo de software de gestión son las aplicaciones ERP (Enterprise Resource Planning) que traducido al español quiere decir Planificación de los Recursos de la Empresa.

Un ERP puede contener software para gestión de producción, gestión de clientes, compras, cuentas por pagar, cuentas por cobrar, contabilidad general, facturación, gestión de inventario, recursos humanos, nóminas o cualquier otra función que tenga que desarrollar dentro de la empresa.

1.3.1 Antecedentes de las aplicaciones ERP

Un ERP tiene su origen en el software empleado en entornos industriales. En los años 60, el principal uso del software en entornos industriales era para la gestión de inventario. Porque en aquella época, la mayor parte del software utilizado en estos entornos era diseñado por los conceptos tradicionales de gestión de inventarios.

En los años 70 se empezó a prestar más atención al uso del software que en inglés se conoce como MRP (Material Requirement Planning), traducido al español se entiende como Planificación de Necesidad de Materiales. Básicamente, lo que se esperaba de este software es que ayudase a planificar qué materiales se iban a necesitar durante el proceso de producción, así como el estudio de la adquisición de los mismos.

En los años 80, surge MRP (Manufacturing Resources Planning, sus siglas en español significan Planificación de los Recursos para la Producción o Fabricación, que rápidamente evoluciona al MRP-II, incluyendo también la gestión de la planta de fabricación y actividades relacionadas con la distribución de los artículos fabricados.

A principios de los 90, MRP-II fue ampliado aún mas para abarcar áreas como Ingeniería, Finanzas, Recursos Humanos y Gestión de Proyectos; que son la totalidad de las funciones desarrolladas dentro de una empresa. Fue esta evolución lo que introdujo el concepto (y el término) ERP.

1.3.2 Importancia del ERP

Estas aplicaciones se consideran importantes porque ayudan a las empresas a entender mejor su actividad, estandarizar sus procesos de negocios y definir mejores políticas. Las ERP ayudan a crear procesos más eficientes por lo que las empresas se pueden concentrar más en otras tareas, como es el servir a sus clientes y maximizar los beneficios.

Existen seis fabricantes principales de ERP, algunos de ellos son: SAP, Oracle, PeopleSoft, JD Edwards Baan, Siebel.

Ejemplos de algunos Software de ERP

WorkPLAN, Software ERP para la gestión industrial de proyectos: Es una base de datos inteligente, y un sistema de comunicación fácil para los clientes y proveedores. Produce de forma automática informes de calidad en cada proyecto.

WMS (Systore): se trata de un Warehouse Management Software: es escalable y flexible, capaz de satisfacer las necesidades específicas de todo sector y cliente. Su uso permite obtener ventajas en términos de: costos operativos inferiores, mejor nivel del servicio a los clientes propios, uso correcto de los recursos y de las herramientas de almacenamiento y de desplazamiento.

Enraf: tiene como misión ser un proveedor líder a nivel internacional para el suministro de instrumentación de alta calidad, de sistemas de software, de servicios y de soporte para la gestión de almacenamiento a granel y para operaciones asociadas para mejorar la competitividad de sus clientes. Ofrece una gran variedad de sistemas de gestión de inventario bajo el nombre de Entis.

Entis Pro: es el sistema de gestión de inventario altamente fiable y probado sobre el terreno para las operaciones de custodia de transferencia.

Entis XL es el paquete más versátil para satisfacer las necesidades específicas del cliente.

Ambos paquetes incluyen varios módulos para centrarse en los aspectos específicos del control de inventario, transferencia de custodia y movimiento y operaciones de crudo.

A partir de las diferentes utilidades que presentan el software de gestión se puede hablar de la calidad que requieren estos software y para ello es necesario definir algunos conceptos relacionados con la calidad.

1.4 Conceptos de Calidad

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

El American Heritage Dictionary, define la calidad como “Una característica o atributo de algo”. La calidad de un producto es medible a través de estándares como longitud, color, propiedades eléctricas, maleabilidad entre otros. La calidad de un proceso o servicio radica en la satisfacción de las necesidades del usuario/cliente, que teóricamente, se han solicitado.

La ISO 8402 [1994] define la calidad como "totalidad de las características de una entidad que le confieren la aptitud para satisfacer necesidades establecidas o implícitas" esta definición considera una entidad no solamente el producto o servicio que se vende sino también, una persona, una organización, un sistema, en otras palabras la amplía a todo lo que hace la calidad.

La ISO 9000 [2005] plantea que calidad es: “Grado en el que un conjunto de características inherentes cumple con los requisitos.”

Y añade dos notas:

Nota 1. El término “calidad” puede utilizarse acompañado de adjetivos tales como pobre, buena o excelente.

Nota 2. “Inherente” en contraposición a “asignado”, significa que existe en algo, especialmente como una característica permanente.

En este caso la calidad depende de los requisitos que se planteen por los productores y si es cierto que los mismos satisfacen las necesidades de los clientes.

Según las familias de normas ISO 9000, la calidad no es más que la capacidad de un conjunto de características inherentes de un producto, sistema o proceso, para satisfacer los requisitos de los clientes.

En la norma ISO 9000:2005 se establecen una serie de conceptos relacionados a continuación y que son de utilidad para el presente trabajo.

Gestión de la calidad: actividades coordinadas para dirigir y controlar una organización en lo relativo a la calidad. Está presente en todas las etapas del proceso de producción de bienes o de servicios y en ella se incluyen, en el marco del sistema de calidad las actividades siguientes: la política, la planificación, el control, el aseguramiento y el mejoramiento de la calidad.

Sistema de gestión: sistema para establecer la política, los objetivos y para lograr dichos objetivos.

Sistema de gestión de la calidad: es un sistema de gestión para dirigir y controlar una organización con respecto a la calidad.

Aseguramiento de la calidad: parte de la gestión de la calidad dirigida a inspirar confianza en que se han cumplido los requisitos de la calidad.

Control de calidad: es una parte de la gestión de la calidad orientada al cumplimiento de los requisitos de la calidad.

Planificación de la calidad: parte de la gestión de la calidad enfocada al establecimiento de los objetivos de la calidad y a la especificación de los procesos operativos necesarios y de los recursos relacionados para cumplir los objetivos de la calidad

Nota: El establecimiento de planes de la calidad puede estar incluido en la planificación de la calidad.

1.4.1. Calidad de software

Para lograr la obtención de un software con calidad es necesario la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar o ordenar la filosofía de trabajo, en aras de lograr una mayor facilidad de prueba, y a la vez se eleva la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

La política establecida está sustentada sobre tres principios básicos: tecnológico, administrativo y ergonómico.

El principio tecnológico define las técnicas a utilizar en el proceso de desarrollo del software.

El principio administrativo contempla las funciones de planificación y control del desarrollo del software, así como la organización del centro de ingeniería de software.

El principio ergonómico define la interfaz entre el usuario y el ambiente automatizado.

La adopción de una buena política contribuye en gran medida a lograr la calidad del software, pero no la asegura. Calidad de software es definida como:

"Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente" [R. S. Pressman (1992).]

"Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario.[Pressman,1998]

"La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario" [IEEE, Std.610-1990].

Para la realización de este trabajo se utiliza la definición brindada por Pressman en 1998.

1.4.2 Gestión de la Calidad de Software

"El aseguramiento de la calidad del software no es más que el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el software va a satisfacer los requisitos dados de calidad" [JLovelle, 1999].

El aseguramiento de la calidad del software está presente cuando se logran: Métodos y herramientas de análisis, diseño, programación y prueba.

Actividades para el aseguramiento de la calidad del software.

- Métricas de software para el control del proyecto.
- Verificación y validación del software a lo largo del ciclo de vida.

El aseguramiento de la calidad incluye las pruebas, los procesos de revisión e inspección y la gestión de la configuración del software.

Las pruebas se software no son lo mismo que el aseguramiento de calidad de software, son una parte del proceso de aseguramiento de calidad. Realizar pruebas a un sistema no significa necesariamente que el proceso de desarrollo esté asegurado y tampoco que de manera directa esté mejorando, pero implementar un proceso de pruebas de software, es un buen inicio, para realizar un mejoramiento del proceso de desarrollo basado en los lineamientos del aseguramiento de calidad de software.

Como parte del aseguramiento de la calidad de los productos de software dentro de las empresas productoras existen modelos que son muy eficientes para velar por la calidad, por ejemplo, CMMI (Capability Maturity Model® Integration) que estudia los procesos de desarrollo de software de una organización y produce una evaluación de la madurez de la organización

Uno de los métodos más empleados durante muchos años para garantizar la calidad del software fue el CMM creado por el SEI (Software Engineering Institute), sus siglas en ingles significan Capability Maturity Model. El objetivo de este método es mejorar la calidad del software mejorando la calidad de los procesos utilizados en su desarrollo. Para conseguirlo se apoya en conceptos como la gestión de calidad, la implementación de procesos repetibles, la recopilación de datos estadísticos sobre elementos como tasas de fallos, y el trabajo a nivel de proceso.

El método CMM es definido por Mark C. Paulk como “una aplicación de sentido común de los conceptos de gestión de procesos y mejora de la calidad al desarrollo y mantenimiento del software”

Como todo en el mundo cambia, las metodologías también lo hacen, y el CMM no se ha quedado atrás, se ha ampliado y ahora ha aparecido CMMI que es una evolución de CMM y que integra los siguientes modelos de calidad.

- Capability Maturity Model for Software (SW-CMM) v2.0 draft C.
- Electronic Industries Alliance Interim Standard (EIA/IS) 731_ Integrated Product.

- Development Capability Maturity Model (IPD-CMM) v0.98.

El CMMI establece una escala de cinco niveles para lograr la madurez de la calidad del software, estos son definidos de la siguiente manera, según JGarcía, 2005.

Nivel 1: Es el inicial, donde se debe establecer entre otras cosas un plan formal de Aseguramiento de la Calidad del Software así como la organización general del proyecto.

Nivel 2: Para obtener nivel 2 de CMMI o Procesos Repetible (Gestionado) donde las pruebas hacen énfasis en las áreas de alto riesgo, el plan de Aseguramiento de la Calidad del Software y el plan del desarrollo del software son compatibles. Se deben realizar las siguientes tareas:

- Gestión de requisitos
- Planificación de proyectos
- Seguimiento y control de proyectos
- Gestión de proveedores
- Aseguramiento de la calidad
- Gestión de la configuración

Nivel 3: Proceso Definido es donde se trabaja en las Pruebas del Software que constituyen un elemento crítico para la calidad del software. Los procesos que hay que realizar para alcanzar este nivel en CMMI son:

- Desarrollo de requisitos
- Solución Técnica
- Integración del producto
- Verificación
- Validación
- Desarrollo y mejora de los procesos de la organización
- Definición de los procesos de la organización

- Planificación de la formación
- Gestión de riesgos

Análisis y resolución de toma de decisiones

Nivel 4: Cuantitativamente Gestionado, es el del Proceso administrativo

- Se puede seguir con indicadores numéricos (estadísticos) la evolución de los proyectos.
- Las estadísticas son almacenadas para aprovechar su aportación en siguientes proyectos.
- Los proyectos se pueden pedir cuantitativamente.

Nivel 5: Optimizado

Se basa en criterios cuantitativos se pueden determinar las desviaciones más comunes y optimizar procesos.

El modelo CMMI es eficaz para las instituciones porque estudia que la madurez de un proceso, es un indicador que mide la capacidad para construir un software de calidad. Es un modelo para la mejora de las organizaciones. Obliga a una revisión constante.

La gestión de la calidad es el aspecto que determina y aplica la política de la calidad, los objetivos y las responsabilidades, y lo realiza utilizando medios como la planificación de la calidad, el control de la calidad, la garantía de calidad y la mejora de la calidad.

La gestión de la calidad es responsabilidad de todos los niveles ejecutivos, pero debe estar guiada por la alta dirección. Su realización involucra a todos los miembros de la organización.

La norma ISO 8402 define como sistema de gestión de la calidad: “Conjunto de la estructura de la organización, de responsabilidades, procedimientos, procesos y recursos que se establecen para llevar a término la gestión de calidad”.

Desde la década del 70, la calidad del software ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software, los

cuales han realizado gran cantidad de investigaciones al respecto con dos objetivos fundamentales basados en como lograr un software con calidad y de cómo evaluar la calidad del software.

Estos objetivos están estrechamente ligados, con la calidad del software, que se refiere a eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad. Todas las metodologías y herramientas tienen un único fin, producir software de gran calidad.

Es importante destacar que las pruebas que se les realiza al software da el resultado del trabajo realizado en equipo y una vez probado el software es donde sale a la luz si durante el desarrollo del mismo se trabajó con eficiencia y calidad.

1.5 Fundamentos y principios de las pruebas de Software

Al analizar los conceptos dados por diferentes autores de que es la calidad en el software, los que más se acercan al significado de prueba son G. J. Myers, que planteó: "el test es el proceso de ejecutar programas con la intención de encontrar errores", y E. W. Dijkstra, quien lo define como "el test puede mostrar la presencia de errores (bugs), pero nunca su ausencia".

Las pruebas de software tienen su principal fundamento en la revisión final de las especificaciones, del diseño y de la codificación, y tiene como objetivo encontrar el mayor número de errores posibles con el menor costo en tiempo y esfuerzo, para esto se puede afirmar que un buen caso de prueba es aquel que tiene una alta probabilidad de detectar errores y posteriormente mostrarlo.

Los principios que se debe seguir en un proyecto al cual se les van a realizar las pruebas son los siguientes:

- Las pruebas deben planificarse mucho antes de que se empiece a desarrollar la programación, esta debe de iniciarse en cuanto se haya terminado la lista de requerimientos del sistema.
- Las distintas pruebas que se les hace al software en cualquiera de las iteraciones de las fases de elaboración, construcción y transición deberán ser un seguimiento de

los requerimientos del cliente/usuario, porque los errores más graves para el cliente son aquellos que impiden cumplir con los requerimientos.

- Las pruebas deberán de planificarse antes de que empiecen los programadores a desarrollar el software y deben empezar cuando esté completo el modelo de requerimientos (antes de ejecutar el código).

  Las pruebas deberían empezar de lo individual a lo general:

  - Módulos individuales
  - Grupos de módulos integrados
  - Sistema total.

  Para hacer más efectivas las pruebas, lo principal es organizar dentro del proyecto en desarrollo un equipo que sea encargado de pruebas dentro del equipo del proyecto porque no es efectivo que los mismos que desarrollen el software sean partes del equipo de prueba.

  Los casos de prueba deben ser diseñados de forma tal que incluyan tanto entradas inválidas, como condiciones válidas. Para el desarrollo de las pruebas es fundamental definir las diferentes estrategias de prueba a utilizar para desarrollar los casos de pruebas a un software.

1.5.1 Estrategias utilizadas en el diseño de casos de pruebas

Las estrategias de pruebas del software integran las técnicas de diseño de casos de prueba. Pressman define los casos de pruebas como “un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo en particular” en una serie de pasos bien planificados que llevan a la construcción correcta del software. Estos pasos son los siguientes [Pressman, 2000]:

- La prueba comienza en el nivel de módulo y trabaja “hacia afuera”.
- En diferentes puntos son adecuadas a la vez distintas técnicas de prueba.
- La prueba la realiza la persona que desarrolla el software y para grandes proyectos, un grupo de pruebas independiente.
- La prueba y la depuración son actividades diferentes.

En la actualidad se calcula que los procesos de prueba representan más de la mitad del costo de un programa, ya que requiere de un tiempo similar al de la programación, lo que obviamente presenta un alto costo económico, suele superar el 80%, siendo esta etapa más cara que el propio desarrollo y diseño de los distintos programas que conforman el sistema.

Un proceso de pruebas requiere mucho más que tiempo y dinero, necesita una verdadera metodología, la cual exige herramientas y conocimientos destinados a dicha tarea.

Existen diferentes tipos de pruebas para realizar a un software en construcción como son las pruebas de sistema, prueba de regresión, pruebas de unidad, entre otras. Esta investigación se concentra en las estrategias utilizadas para probar al software después de la programación, teniendo en cuenta la repercusión que han tenido en el mundo. Cuando se considera que un módulo está terminado se les realizan las pruebas sistemáticas, con el objetivo de buscar fallos a través de un criterio específico, estos criterios se denominan "pruebas de caja negra y de caja blanca".

1.6 Diseño de Pruebas

El objetivo técnico del Diseño de Pruebas es garantizar con el mayor grado de confianza posible que se detectarán los defectos del software y equilibrar entre recursos y garantía para descubrir los defectos existentes.

Existen tres enfoques principales para el diseño de casos de prueba.

1. El enfoque funcional o de caja negra
2. El enfoque estructural o de caja blanca
3. El enfoque aleatorio, consistente en utilizar modelos (en muchas ocasiones estadísticos) que representen las posibles entradas al programa para crear a partir de ellos los casos de prueba.

Se explicaran los dos primeros enfoques.

Métodos de prueba

Existen diversos métodos para realizar las pruebas de software, entre los más importantes se encuentran la prueba de Caja Blanca, y la Prueba de Caja Negra. La prueba de caja blanca es la mejor de su tipo para verificar que se recorran todos los caminos y detectar un mayor número de errores. Mientras que la prueba de caja negra brinda la posibilidad de cubrir la mayor parte de las combinaciones de entradas y lograr con ello un juego de pruebas más eficaz.

Pruebas de Caja Negra

Las pruebas de caja negra son aquellas que se enfocan directamente en el exterior del módulo o software, sin importar el código, son pruebas funcionales en las que se trata de encontrar fallas en las que este no se atiene a su especificación, se centra en como relacionar la interfaz con el usuario, apariencia de los menús, control de las teclas, etcétera. Este tipo de pruebas no es aplicable a los módulos que trabajan en forma transparente al usuario.

Para realizar estas pruebas existe varias técnicas pero la mas usada es la técnica algebraica llamada "clases de equivalencia", que consiste en tratar a todas las posibles entradas y parámetros como un modelo algebraico y utilizar las clases de este modelo para probar un amplio rango de posibilidades.

Estas pruebas permiten detectar:

- funcionamiento incorrecto o incompleto del sistema
- errores en la interfaz
- problemas de rendimiento
- errores de inicio y terminación

La técnica de Caja negra es la más usada por las características que tiene de cómo sin entrar en la parte del código se puede probar la usabilidad del sistema y comprobar si los requisitos del cliente son puestos explícitamente. Esta técnica es la que actualmente se está utilizando en la UCI, y por las diferentes ventajas que posee se propone usarla en este proyecto de Sistema de Gestión de Inventarios y Almacenes.

Pruebas de Caja Blanca

Las pruebas de caja blanca son mucho más amplias, también se denominan pruebas de cobertura o pruebas de caja transparente, indica cuánto código del programa se ha probado. Básicamente la idea de pruebas de cobertura consiste en diseñar un plan de pruebas en las que se vaya ejecutando sistemáticamente el código hasta que haya corrido todo o la gran mayoría de él.

Este tipo de prueba se usa con frecuencia debido a su gran importancia pero a la vez se realizan a software donde los clientes tienen un nivel mucho más avanzado y están interesados en probar la estructura interna del sistema, según entrevistas realizadas a diferentes desarrolladores de prueba en la universidad, se pudo concluir que no se aplican pruebas de caja blanca y es por este motivo que se propone una estrategia para desarrollar las prueba de caja blanca al software Sistema de Gestión de Inventario de Almacenes.

La prueba de la caja blanca, a primera vista, podría parecer impracticable puesto que no es posible aplicarla exhaustivamente para grandes sistemas, sin embargo no se debe desechar; ya que se puede elegir y ejercitar una serie de caminos lógicos importantes, que tienen además las estructuras de datos más importantes para comprobar su validez.

Se pueden combinar ambos métodos para llegar a un método que valide la interfaz del software y asegure selectivamente que el funcionamiento interno del software es correcto.

1.6.1 Método de Prueba de Caja Blanca

La prueba del camino básico es una técnica de prueba de caja blanca propuesta por Tom McCabe en 1976. Esta prueba permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

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

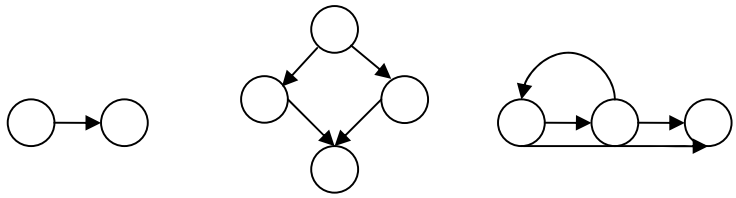
Para aplicar la técnica del camino básico se debe introducir la notación para la representación del flujo de control, este puede representarse por un Grafo de Flujo en el cual:

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.

Se calcula la complejidad ciclomática del grafo.

Para construir el grafo se debe tener en cuenta la notación para las instrucciones. Esta puede observarse en las figuras 2 y 3.



Secuencia **If** **While**

Figura 2. Notación de grafos de flujo para las instrucciones: Secuenciales, If, While.

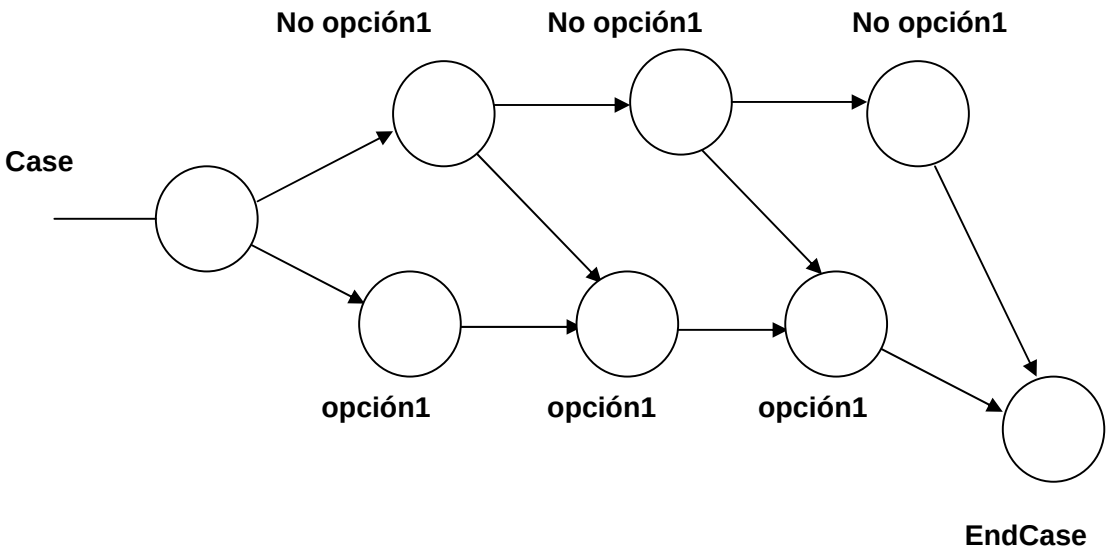


Figura 3. Notación de grafos de flujo para la instrucción Case.

Un grafo de flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, su comprensión y brindan información para confirmar que el trabajo se está haciendo adecuadamente.

Componentes del grafo de flujo

- 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.

En la figura 4 se observa la representación un grafo de flujo, en el cual aparecen sus componentes.

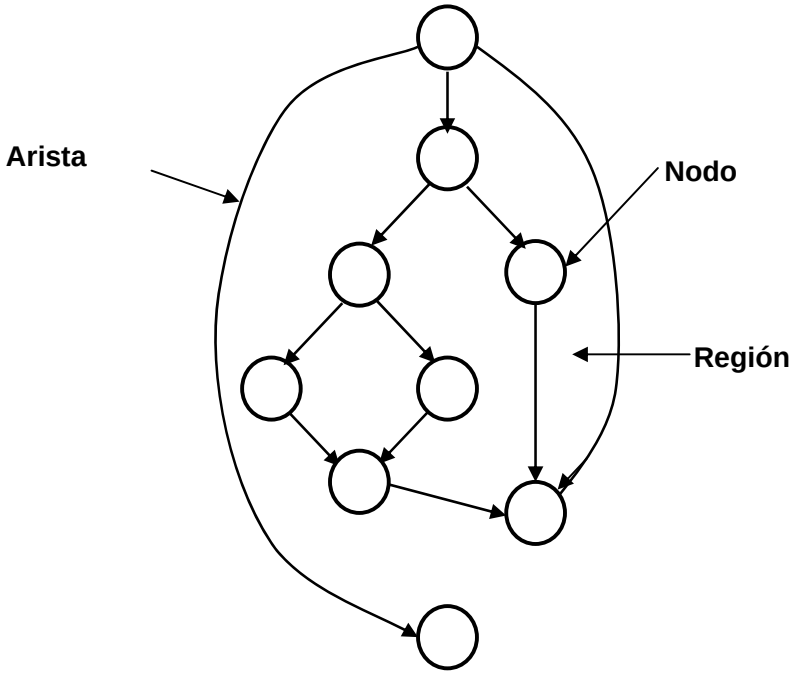


Figura 4. Componentes de los grafos de flujo

Cálculo de la complejidad ciclomática a partir de un segmento de código

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 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.

Si se diseñan pruebas que fuesen 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.

Formas fundamentales de 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 contenidos en el grafo G.

1.6.2 Método de Prueba de Caja Negra

El diseño de pruebas del Sistema de Gestión de Inventario se basa en una de las técnicas anteriormente expuestas la técnica de Caja negra. Para la utilización de esta técnica se utiliza una plantilla (Ver Anexo #1) donde están los principales parámetro para medir la funcionalidad del Sistema a través de la interfaz.

Esta plantilla está determinada por los siguientes parámetros:

- ✓ Descripción general del caso de uso
- ✓ Tipo de prueba que se va a realizar al caso de Uso
- ✓ Descripción de esa prueba
- ✓ Flujo central del Caso de Uso
- ✓ Condiciones de ejecución
- ✓ Datos sobre la prueba realizada al caso de uso.

En esta tabla se llenan los siguientes campos de la prueba realizada al caso de uso correspondiente a la interfaz de la aplicación.

- Las clases válidas son los datos validos con los cuales funcionan correctamente los campos de la interfaz.

- Clases inválidas son valores que se les entran a los campos de la interfaz, que no son compatibles con los valores reales de la misma, para ver si la aplicación es capaz de devolver un cartel de error con un carácter no válido entrado a la interfaz de la aplicación.
- Resultado esperado: es lo que se espera de la aplicación, su devolución
- Resultado de la prueba: es ver si fue satisfactorio o insatisfactoria
- Las observaciones: es lo que pudo hacerse para que funcionara mejor la aplicación.
- Por ciento de cumplimiento: hasta dónde se cumplió la prueba realizada. En la tabla "Registro de defectos y dificultades detectados" se redacta una lista de no conformidad de cada uno de los elemento probados, que no es más que el incumplimiento de un requisito, señalando la etapa donde se detectó el error. Se hacen recomendaciones para mejorar el trabajo realizado.

Aplicando el diseño de casos de pruebas al software en cuestión se puede conseguir una prueba más completa y descubrir y corregir el mayor número de errores antes de que comiencen las "pruebas del cliente". Para probar se necesita organización y planificación de las pruebas que se van a realizar.

La forma más práctica de aplicar los tipos de pruebas y la que se propone es la siguiente:

- Hacer pruebas de Caja Negra analizando valores límites. Recordando que hay que analizar condiciones límites de entrada y de salida. Estas prueba solo se les realiza a módulos que van a ser interfaz con el usuario, y se apoya en los requisitos funcionales de cada módulo.
- Identificar clases de equivalencia de datos (entrada y salida) y añadir más pruebas de Caja Negra para contemplar valores normales (en las clases de equivalencia en que estos sean diferentes de los valores límites; es decir, en rangos amplios de valores).
- Medir la cobertura de caja blanca que se ha logrado con las fases previas y añadir más pruebas de Caja Blanca hasta lograr la cobertura deseada.

- Diseñar casos de prueba para examinar la lógica del programa para garantizar que se ejerciten todos los caminos independientes de cada módulo, todas las decisiones lógicas, y que se ejecutan todos los bucles y las estructuras de datos internas.

1.7 Herramientas de pruebas

Las herramientas tales como Checking, Test, JTest 8.0 eliminan la complejidad de las pruebas. Estas herramientas de prueba están orientadas al modelado y al diseño de las aplicaciones, y al análisis estático del código.

ChecKing es la herramienta de monitorización del proceso de desarrollo del software y sus resultados cubren las necesidades de las organizaciones que desean controlar la calidad del software antes de llegar a manos del cliente. Para ello la herramienta sigue un modelo de métricas en el que se integran medidas obtenidas automáticamente del proceso de desarrollo (actividad, requisitos, defectos y cambios) y de elementos analizables del software: código fuente, documentación del proyecto, scripts de construcción, y scripts de pruebas.

TEST es una herramienta líder en automatización de pruebas unitarias y de productos con código estándar, trabaja sobre clases escritas en la plataforma Microsoft.NET, sin requerir que los desarrolladores realicen un solo caso de prueba o stub. Mejora la fiabilidad, funcionalidad, seguridad, desarrollo y mantenimiento de las aplicaciones .NET. Trabaja con lenguajes de programación que utilizan el marco Microsoft .NET, incluyendo C# y VB.NET. Puede probar cualquier fichero o paquete que haya sido construido utilizando el CLR de .NET. Facilita la creación y ejecución de pruebas definidas por el usuario basado en el framework de NUnit.

JTEST 8.0 ofrece varios avances para las industrias desarrolladoras de software de alta calidad. Estos avances tecnológicos están concentrados en la realización de pruebas, para ayudar a los equipos a verificar de manera automática la funcionabilidad de aplicaciones cada vez más complejas, en empresas con sistemas en permanente cambio (J2EE, servicios de SOA/Web), todo esto para generar un incremento en la satisfacción del cliente. Jtest permite reducir el tiempo de entrega del software así como una disminución del riesgo de generar software defectuoso o con problemas de vulnerabilidad.

Para reducir la complejidad de las pruebas, Jtest ofrece la generación y la ejecución automatizada de los casos de prueba a través de la simulación de un entorno real, posibilita una prueba en tiempo de ejecución que permite la temprana detección de defectos en el código que de otro modo pasarían inadvertidos hasta etapas avanzadas de desarrollo como el aseguramiento de la calidad.

La herramienta Jtest posee diferentes características tales como:

- prueba métodos individuales, clases, o grandes y complejas aplicaciones.
- soporta Struts, spring, Hibernate, EJBs, JSPs, servlets.
- genera casos de prueba funcionales JUnit que capturan el comportamiento real del código.
- genera casos de prueba JUnit y Cactus que muestran la fiabilidad y capturan el comportamiento.

Parametriza los casos de prueba para usarlos con múltiples valores (generados en tiempo de ejecución, definidos por el usuario o provenientes de otras. fuentes de datos).

1.8 Metodologías de desarrollo de Software

A continuación se analiza el concepto de trazabilidad de un caso de uso al cual se les realizaran pruebas, que es definido como:

La trazabilidad es un conjunto de medidas, acciones y procedimientos que permiten registrar e identificar cada producto desde su origen hasta su destino final.

Consiste en la capacidad para reconstruir la historia, recorrido o aplicación de un determinado producto, identificando:.

Origen de sus componentes.

Historia de los procesos aplicados al producto.

Distribución y localización después de su entrega.

Para poder desarrollar el producto que se debe entregar al cliente es necesario utilizar diferentes metodologías de desarrollo de software con el fin de poder

garantizar la calidad del mismo. Dentro de las metodologías más utilizadas para el desarrollo de software a nivel mundial se encuentran la XP (Extreme Programing), MSF (Microsoft Solution Framework) y RUP (Rational Unified Process),

Una metodología de desarrollo no es más que un conjunto de actividades que tienen un orden lógico y que garantizan que al finalizar la ultima actividad de la metodología se tiene el software que se comienza a analizar en la primera actividad con la utilización de herramientas y técnicas.

Metodología XP (Extreme Programing)

Es una metodología exitosa de desarrollo de software, en la actualidad es utilizada para proyectos de corto plazo, de pocos integrantes en el equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Metodología MSF (Microsoft Solution Framework)

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

RUP (Rational Unified Process),

El “Proceso Unificado” es el resultado final de tres décadas de desarrollo y uso práctico. Esta es una de las causas que conlleva a que sea la metodología que mejor se ajusta a las necesidades que existen actualmente en el desarrollo de software, pues propone un Modelo iterativo e incremental, centrado en la arquitectura y guiado por casos de uso muy acorde con la naturaleza cambiante de los requisitos en muchos proyectos. Es una metodología que está totalmente respaldada por una excelente herramienta CASE: Rational Rose.

Esta metodología se divide en 4 fases:

- Inicio: el objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración: en esta etapa el objetivo es determinar la arquitectura óptima.

- Construcción: en esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- Transmisión: el objetivo es llegar a obtener el reléase del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

RUP como ya se había mencionado anteriormente es una metodología iterativa incremental, que va eliminando los errores cometidos en las iteraciones previas, logrando que al final del proceso se obtenga como resultado un producto con calidad. Define los roles a jugar por cada miembro del equipo de desarrollo dentro de los que se encuentra uno de los roles más importante, el Ingeniero de prueba.

1.8.1 Rol del ingeniero de prueba

El objetivo principal del ingeniero de prueba consiste en el desarrollo y ejecución de casos y escenarios de prueba, mediante el uso de metodologías formales de pruebas, así como pruebas de regresión y el correspondiente reporte basándose en la definición de requerimientos, documentación técnica y en coordinación con los analistas de los procesos de negocio y de desarrollo de la aplicación. Asimismo, el analista de calidad del software se encargará de la gestión de los errores hasta su resolución.

Entre sus responsabilidades se incluyen el diseño de los casos de pruebas, análisis de los problemas seguimiento de problemas y de los resultados, que incluye la validación de bugs, y generación de informes de progreso.

El ingeniero de prueba tiene las siguientes responsabilidades: Trabajar con el equipo de desarrollo para validar los requerimientos para resolver incidencias de diseño y defectos de software.

Establecer los juegos de pruebas. Ejecutar los planes de pruebas en los entornos de integración y sistema de la nueva release y de regresión para asegurar la integridad de la aplicación.

Desarrollar y mantener los juegos de pruebas y escenarios de prueba basándose en los requerimientos del cliente y el diseño.

- Preparar los juegos de datos necesarios.
- Ejecución de planes de prueba detallados.
- Ejecutar los casos de test de acuerdo con la metodología de pruebas definida.

- Asegurar la ejecución de las actividades de test siguiendo los planes de proyecto y los hitos definidos.
- Dirigir el análisis detallado de los resultados de las pruebas tanto correspondiente a pruebas manuales como a pruebas automáticas.
- Proporcionar retroalimentación de los resultados de las incidencias en las pruebas.
- Colaborar con el equipo de desarrollo para resolver los defectos encontrados.
- Investigar y proponer métodos y estándares desarrollando scripts de test, juegos y escenarios de test.
- Llevar a cabo las pruebas de integración, test funcional, aceptación de usuario y de interfaz de usuario.
- Crear informes resumen de pruebas que comuniquen los resultados de las pruebas con claridad.

1.9 Plan de Prueba

El propósito del plan de pruebas es explicitar el alcance, enfoque, recursos requeridos, calendario, responsables y manejo de riesgos de un proceso de pruebas.

Un plan de pruebas está constituido por un conjunto de pruebas. Cada prueba debe:[J.A. Mañas, 2002].

- ✓ Dejar claro qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad,etc.)
- ✓ Dejar claro cómo se mide el resultado

- ✓ Especificar en qué consiste la prueba (hasta el último detalle de cómo se ejecuta).
- ✓ Definir cuál es el resultado que se espera (identificación, tolerancia, etc.)
- ✓ Cómo se decide que el resultado es acorde con lo esperado

Las pruebas carecen de utilidad si no se sabe exactamente lo que se quiere probar, si no está claro cómo se prueba, o si el análisis del resultado se hace a simple vista.

Un caso de prueba consta de 3 bloques de información:

1) El propósito de la prueba

2) Los pasos de ejecución de la prueba

3) El resultado que se espera

Cada uno de esos puntos debe quedar perfectamente documentado.

El plan de prueba recoge y señala el enfoque que se le va a dar a las pruebas, los recursos que se utilizarán y el esquema de actividades de prueba, así como los elementos a probar, las características, las actividades de prueba, el personal responsable y los riesgos asociados.

Especifica las propiedades necesarias y deseables del ambiente de prueba, incluyendo las características del hardware, el software de sistemas, herramientas de soporte, configuración de entornos de prueba y recursos humanos

1.10 Necesidad del diseño de pruebas para el Sistema de Gestión de Inventario y Almacenes.

Es importante diseñar las pruebas para el SIGIA porque éstas constituyen un elemento crítico para la garantía de la calidad del sistema que se está desarrollando, y representan una revisión final de las especificaciones, del diseño, y de la codificación.

Las ventajas que trae la realización de pruebas para el desarrollo de este sistema son las siguientes:

- Reducen la posibilidad de agregar defectos al software. Si hay que adicionar una característica requerida por el cliente y se prueba que ya no funcionan correctamente algunas de las cosas que anteriormente funcionaban, se puede inferir que la nueva funcionalidad introducida es la que contiene defectos, por lo que no hay necesidad de realizar modificaciones a los componentes realizados anteriormente.
- Reducen la posibilidad de encontrar defectos en funcionalidades ya implementadas.
- Reducen el costo del cambio. Evitan que se descubran los defectos hasta el final del desarrollo del software.
- Permiten verificar si se cumplen los requisitos que el cliente esperaba del software.
- Con las pruebas el programador sabe la funcionalidad del software que tiene que programar.

1.11 Conclusiones parciales

- El aseguramiento de la calidad del software permite reducir notablemente los costos de producción e implantación, y proporciona mayor confianza en el cumplimiento de los requisitos del cliente.
- Las pruebas de software son un elemento imprescindible para asegurar la calidad y para verificar el cumplimiento de los requisitos funcionales y los impuestos por el cliente.
- El rol del ingeniero de prueba es muy importante dentro del equipo de desarrollo de un software porque es el que verifica la eficacia del trabajo realizado y el cumplimiento de los requisitos impuestos por el cliente.
- La técnica de Caja negra es la más usada por las características que tiene de cómo sin entrar en la parte del código se puede probar la usabilidad del sistema y comprobar si los requisitos del cliente son puestos explícitamente. Esta técnica es la que actualmente se está utilizando en la UCI.

CAPITULO 2 DISEÑO DE CASOS DE PRUEBAS PARA EL SISTEMA DE GESTIÓN DE INVENTARIOS Y ALMACENES

2.1 Introducción

Las tecnologías de la Información y las Comunicaciones (TIC) han traído múltiples consecuencias para el manejo, control y preservación de la información asociada a cualquier entidad que preste servicios o venda productos.

La empresa cubana no puede sustraerse a esta corriente de innovación, pues aunque se sigan usando eficazmente los medios tradicionales y estos demuestren su eficiencia, las nuevas tecnologías con todas las ventajas que de su uso se derivan complementan dichos medios asegurando su futuro. El Ministerio de la Informática y las Comunicaciones (MIC), junto a otras importantes entidades, en su afán por propiciar el desarrollo de nuestro país, protagoniza un proceso de informatización de la sociedad que alcanza los principales objetivos económicos y no está exento de esto el proceso de inventario y aseguramiento de los recursos materiales en las empresas del país. Con el adelanto tecnológico, es preciso garantizar la calidad del software y para esto la prueba es una de las medidas más importantes, puesto que su objetivo principal es detectar errores en el software. Es decir, demuestran la carencia de la calidad, revelada por los defectos encontrados. Por consiguiente, la realización de las pruebas es un requisito previo necesario para construir e implementar satisfactoriamente un sistema. Sin embargo, en muchas organizaciones, se encuentran en una etapa experimental. Un ejemplo es la Universidad de las Ciencias Informáticas, joven aún en la actividad de probar software y es por eso que solo realiza pruebas de caja negra a todo el software desarrollado en ella.

Los objetivos de este capitulo son:

Diseñar un plan de prueba para establecer los pasos a seguir en la realización de los casos de prueba para el Sistema de Gestión de Inventario de Almacenes (SIGIA).

Diseñar cada uno de los casos de pruebas para los módulos que componen el producto SIGIA para la integración del plan de prueba.

Realizar un análisis de los resultados obtenidos en las pruebas realizadas a cada uno de los casos de uso de los módulos nomencladores y administración del Sistema de Gestión de Inventario.

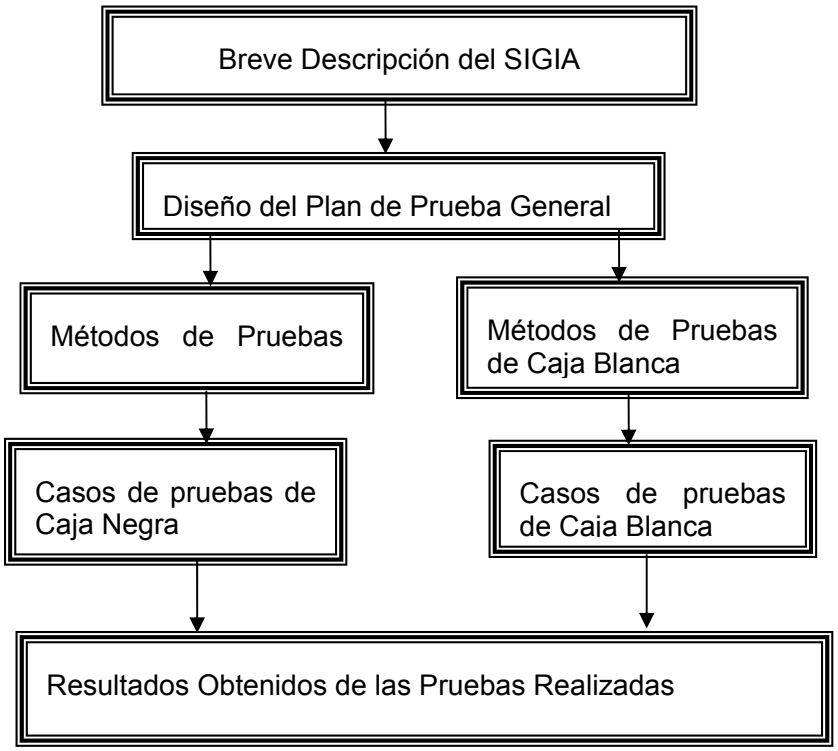


Figura 5. Hilo Conductor del capítulo 2

2.2 Sistema de Gestión de Inventario de Almacenes

El control y la gestión de almacenes son tareas medulares de las que depende en gran medida el éxito o fracaso de un negocio donde se manejen recursos materiales. El control de la mercancía almacenada de manera precisa y organizada permite reducir los gastos por concepto de locales y materiales y aumentar las rotaciones de los productos en caso de que estos se vendan, lo que es un importante indicador de la rentabilidad económica. La Gestión de Inventarios regula todo lo relativo al control y disposición de las existencias de determinados bienes, en la cual se aplican métodos y estrategias que pueden hacer rentable y productivo

la tenencia de los mismos y a la vez sirve para evaluar los procedimientos de entradas y salidas de dichos productos.

En nuestro país se han desarrollado numerosos sistemas para la gestión de almacenes, aunque la mayoría de estos están especializados para el funcionamiento de determinada empresa o tipo de empresas. Ejemplos de ellos los constituyen DRIM e IHMM, este último desarrollado con el objetivo de usarse en instalaciones hoteleras. La ausencia de una aplicación que integrara todas las funcionalidades independientemente del tipo de empresa que haga uso de ella sustentó la decisión de construir una alternativa con el fin de estandarizar el proceso de gestión de inventarios y almacenes, un software configurable a partir de las necesidades particulares de los clientes y que incorporara todas las regulaciones que establece el Ministerio de Finanzas y Precios. Para el desarrollo del mismo fue convocada la Universidad de Ciencias Informáticas (UCI). Una corporación que se ha acercado a algunas funcionalidades que debe tener el nuevo Sistema de Gestión de Inventario ha sido Cubalse. El nuevo Sistema de Gestión de Inventario de Almacenes que se está desarrollando permite:

- ✓ El control de los activos de una empresa,
- ✓ Automatiza las funcionalidades principales de un Sistema de Inventario y Almacén
- ✓ Automatiza la gestión de las compras,
- ✓ Automatiza la generación de información actualizada de las entidades.
- ✓ Presenta una nueva interfaz, más amigable y fácil de usar por los clientes.
- ✓ Permita el uso de envío de documentos vía sistema de transmisión de datos (e-mail u otro).
- ✓ Tiene una seguridad en las Funcionalidades referentes a Administración y seguridad de la Información del Sistema separando a los usuarios por roles.

Para el desarrollo de este software se selecciona un grupo de estudiantes capacitados con el objetivo de garantizar el desarrollo efectivo del Sistema Gestión de Inventarios y Almacenes.

Uno de los factores que afecta la construcción de cualquier sistema y en especial a este, es la calidad del software y específicamente las pruebas pues a menudo se carece de la organización requerida para realizar las pruebas al software. En este sentido, aunque la calidad se toma en serio en cualquier producto, las pruebas no se introducen sino hasta el final del proceso de desarrollo del software y esto es un problema que debe ser solucionado. Al Sistema Gestión de Inventarios y Almacenes, con el objetivo de probar su eficiencia, se le realizarán pruebas de Caja negra y Caja blanca a los módulos de Administración y al módulo de Nomenclar. Para ello se ha trazado un plan de pruebas que garantice el aseguramiento de la calidad del producto.

2.3 Diseño del Plan General de Pruebas

Introducción

El plan de prueba describe las estrategias, recursos y planificación de las pruebas.

Tiene como **objetivo** el establecimiento de un cronograma de pruebas para el SIGIA, así como el diseño de todos los casos de pruebas a aplicar.

Propósito

Tiene como propósito principal lograr hacer un sistema que cumpla con las expectativas del cliente, que este quede satisfecho con la entrega y que el software tenga una documentación entendible para lograr una mejor mantenibilidad del producto y un mejor soporte. En concreto define los siguientes pasos a seguir:

- Identifica los elementos que se van a probar.
- Describe la estrategia de pruebas que se va a seguir en el proceso de prueba.
- Identifica los recursos necesarios para llevar a cabo el proceso de prueba y estima los esfuerzos que lleva.
- Lista los resultados que se obtienen de las actividades de la prueba.

Ámbito

Este Plan de Pruebas describe las pruebas de unidad, integración, de interfaz de usuario, de unidad y de funcionalidad, que se proponen aplicar al Sistema Gestión

de Inventarios de Almacenes u otro sistema en desarrollo. Se prueban todos los requisitos definidos en la **Especificación de requisitos.**

Descripción de los pasos a seguir en las pruebas de Caja Negra y Caja Banca.

Para la realización de este trabajo se realizan pruebas de unidad al módulo de nomencladores, teniendo en consideración que es el módulo que se va a terminar y que son los casos de uso más importantes. En aras de lograr la culminación del proceso de prueba del módulo se decide aplicar las pruebas de Caja Blanca al módulo de nomencladores y Caja Negra a los módulos de nomencladores y administración.

En ambos tipos de prueba se especificó para cada caso, cuál es el resultado esperado durante la ejecución del software en esa sección, con ello se decide si eran realmente esos los resultados que se tenían previstos y por lo tanto detectar la ocurrencia o no de errores en el mismo. Para diferenciar ambas pruebas e identificar el modulo se empleó la siguiente notación.

Método de prueba (MP): Caja Blanca (CB) o Caja Negra (CN)

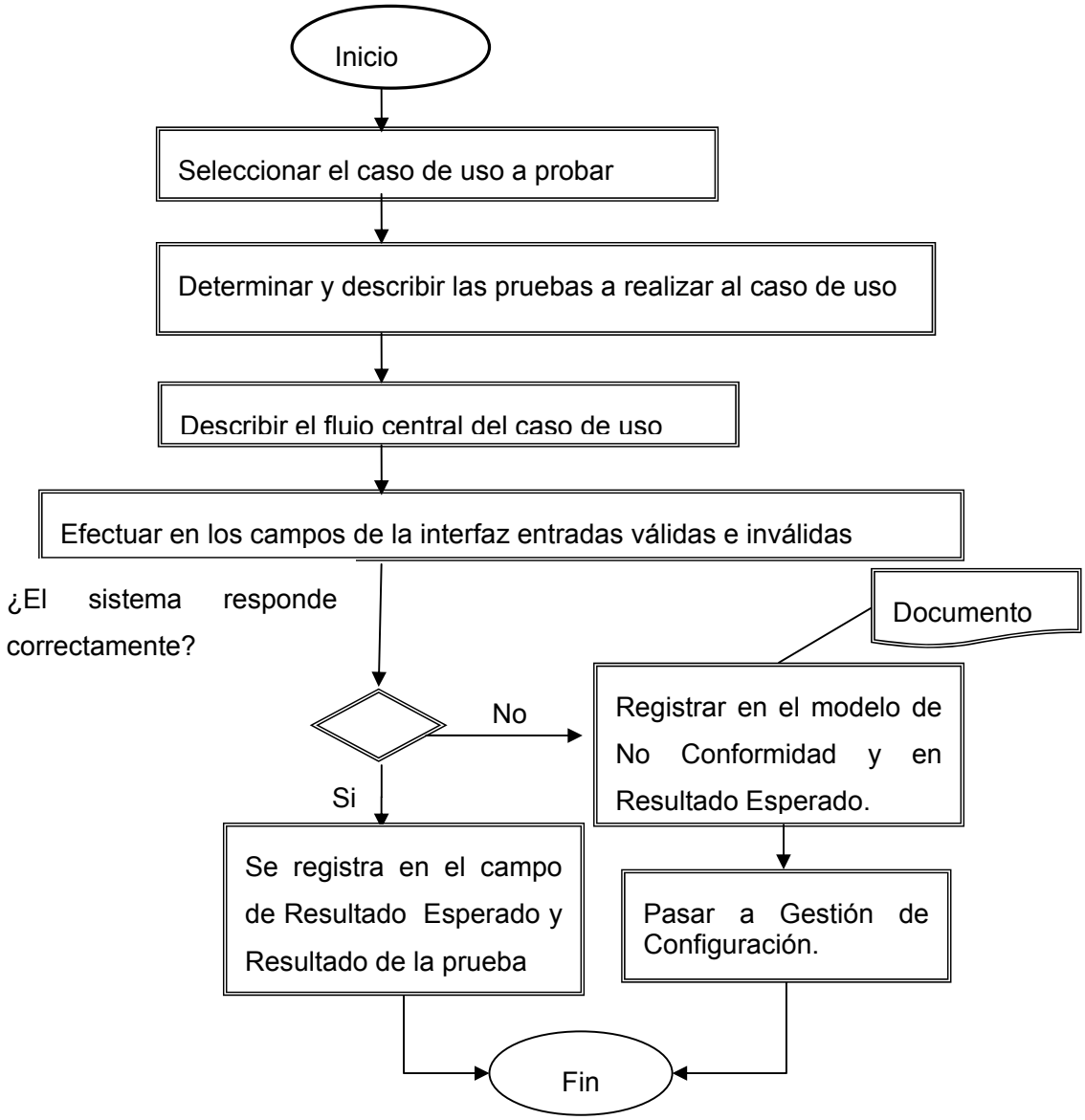
Caso de Prueba: CP

Nombre del módulo: NM

Expresión final: CP- MP-NM

Prueba de Caja Negra

En las pruebas de Caja Negra, la realización de los casos de pruebas se efectúa para todas las entradas del mismo, siguiendo las secuencias de pasos de este **Diagrama de actividades de Caja Negra:**

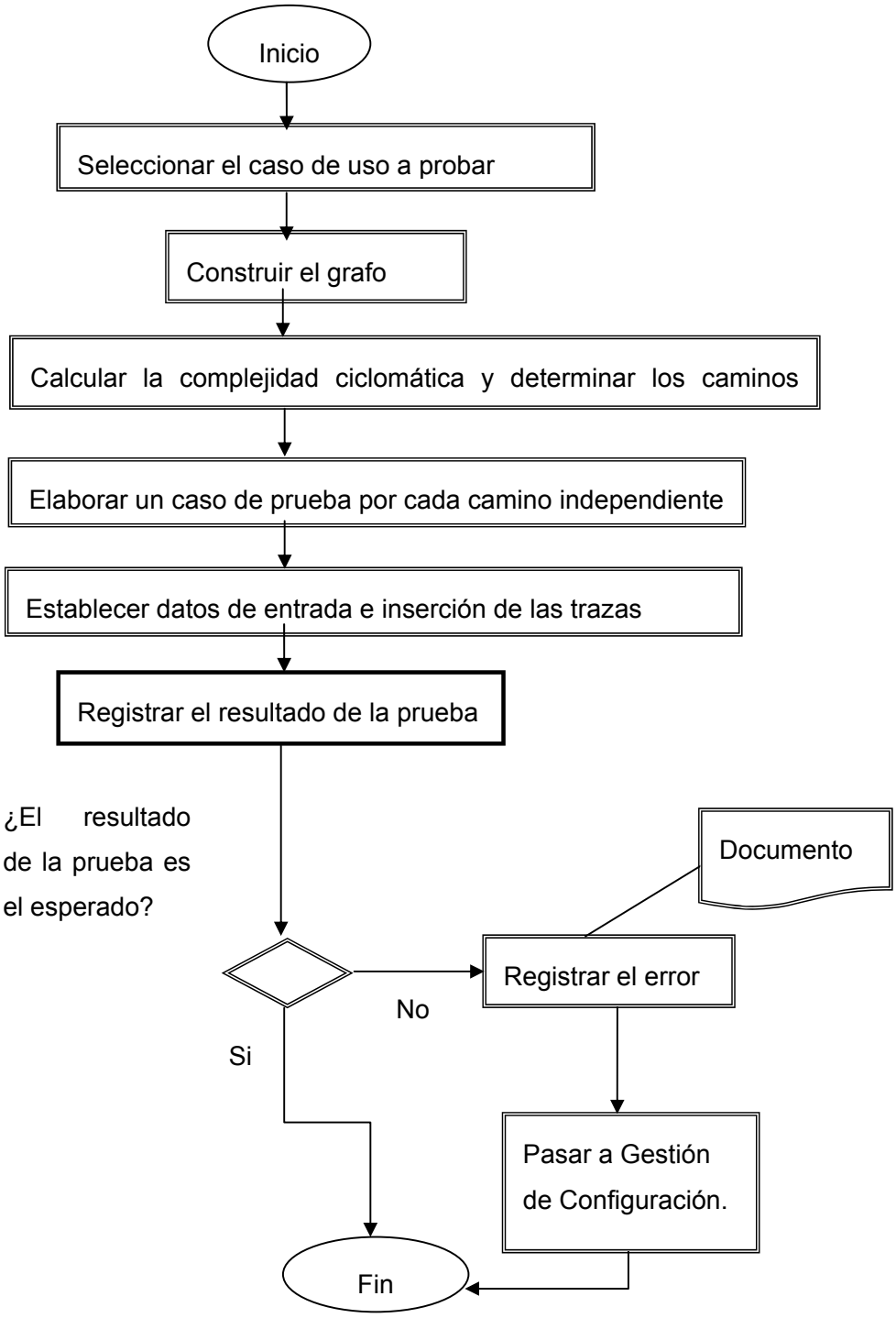

Inicio
Seleccionar el caso de uso a probar
Determinar y describir las pruebas a realizar al caso de uso
Describir el fluio central del caso de uso
Efectuar en los campos de la interfaz entradas válidas e inválidas
¿El sistema responde correctamente?
No
Si
Documento
Registrar en el modelo de No Conformidad y en Resultado Esperado.
Se registra en el campo de Resultado Esperado y Resultado de la prueba
Pasar a Gestión de Configuración.
Fin

Esta prueba se le efectúa a cada módulo, se realizan los casos de pruebas para las entradas, se obtienen las clases de equivalencia y luego se diseñaron los casos de pruebas, en el caso de las clases válidas, que abarcan la mayor cantidad de clases posibles en un caso de prueba, y en caso de las clases inválidas se diseñó un caso para cada una de las diferentes entradas de la interfaz.

Pruebas de Caja Blanca

La prueba de Caja Blanca se efectúa a 5 de las clases de la capa de servicio del módulo nomencladores, pertenecientes a los casos de usos nomenclar Especialidad Proveedor, nomenclar Sección Inventario, nomenclar Unidad de Medida, nomenclar producto, nomenclar proveedor.

Eligiendo para ello los procedimientos de cada clase que presentan las condiciones más complejas (case, for, if), y por ello los más propensos a fallos. Siguiendo la secuencia de pasos del siguiente **Diagrama de actividades de Caja Blanca:**



✓ Primeramente, se obtienen los grafos de flujo, a partir de las instrucciones de código de los procedimientos.

✓ Después de tener confeccionado el grafo con todas las relaciones entre los nodos y aristas, se obtuvieron los caminos básicos, los que constituyen un elemento indispensable para lograr que se recorra el procedimiento en todas sus variantes, finalmente, y como parte concluyente de todo el proceso, se realizan los casos de pruebas para cada uno de los caminos (Diagrama de secuencia de pasos).

✓ El establecimiento de los datos de entradas e inserciones de trazas en todas las aristas que conforman el camino del grafo (Análisis de la necesidad de imprimir algún valor intermedio de las variables significativas del programa).

Requerimiento de las Pruebas

La lista que se muestra en esta sección identifica los elementos (casos de uso, y requisitos no funcionales) que son objetivos de las pruebas de caja negra. Es decir, los elementos que se van a probar.

En la documentación elaborada para el análisis del sistema, se puntualizan las responsabilidades que se deben cumplir para lograr la configuración del software en su etapa de desarrollo. A continuación se muestran algunas de las que debe cumplir el sistema, las cuales están presentes en los 2 módulos implicados en las pruebas.

Pruebas de funcionalidad:

Requisitos de los módulos Administración y Nomenclar (Ver Anexo # 3)

Estrategia de prueba

En esta sección se presenta una propuesta del enfoque que se utilizará para probar cualquier sistema software, y se define cómo se deben realizar las pruebas.

Pruebas de funcionalidad.

Las pruebas de funcionalidad se deben centrar en cualquier requisito que pueda ser trazado directamente de los casos de uso y reglas de negocio. El objetivo de estas pruebas es verificar la aceptación, procesamiento y recuperación de datos y la adecuada implementación de las reglas de negocio. Este tipo de pruebas están basadas en técnicas de caja negra, es decir, verificar la aplicación interaccionando a través de las interfaces de usuario y analizando los resultados.

Objetivos de la prueba	Asegurar la navegación correcta de la aplicación, la entrada de datos, su procesamiento y recuperación.
Técnicas	Ejecutar cada caso de uso y flujo del caso de uso con datos válidos e inválidos para verificar lo siguiente: • Cuando se utilizan datos correctos se obtienen los resultados esperados. • Cuando se utilizan datos incorrectos se obtienen los mensajes de error o advertencias adecuadas. • Cada regla de negocio se ha aplicado correctamente.
Criterios de finalización	Todas las pruebas planificadas se han ejecutado. Todos los defectos identificados se han considerado.

Pruebas de interfaz de usuario.

Las pruebas de interfaz de usuario verifican la interacción del usuario con el sistema software. El objetivo de esta prueba es asegurar que la interfaz de usuario permite al usuario acceder y navegar a través de toda la funcionalidad de la aplicación. Además, la prueba de interfaz de usuario garantiza que las interfaces de usuario cumplen los estándares y para esto se utilizaron las listas de chequeo para dicha prueba.

Objetivos de la prueba	Verificar los siguientes Objetivos: • La navegación a través de la aplicación refleja adecuadamente las reglas de negocio y los requisitos incluyendo ventana a ventana, campo a campo y métodos de acceso (tabulador, movimientos del ratón y teclas de función). • Las ventanas y sus características, como menús, tamaño, posición y estado cumplen los estándares.
Técnicas	Crear o modificar pruebas para cada ventana con el objetivo de verificar la correcta navegación y su estado.

Pruebas de Integración.

La prueba de integración es una técnica sistemática donde se construye la estructura del programa mientras que al mismo tiempo se llevan a cabo pruebas para detectar errores asociados con la interacción.

Objetivos de la prueba	El objetivo es coger los módulos probados en la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.
Técnicas	Las técnicas que más prevalecen son las de diseño de casos de prueba de caja negra para efectuar las pruebas de integración.

Pruebas de Unidad

La prueba de unidad se centra en el módulo. Usando la descripción del diseño detallado como guía, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo.

Objetivos de la prueba	El objetivo de la prueba de unidad es asegurar que cada módulo del código fuente funcione correctamente como una unidad.
Técnicas	Las técnicas que más prevalecen en esta estrategia de prueba son las de Caja Blanca.
Criterios de finalización	Se han completado las pruebas sin ningún error y dentro de los tiempos de respuesta esperados.

Durante la prueba de unidad se realizan las pruebas que se esquematizan en esta **figura 6**

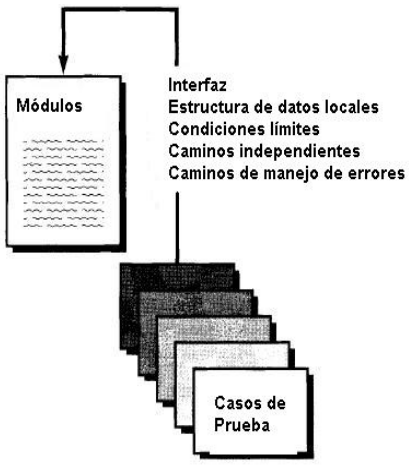


Figura 6. Prueba de Unidad

Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada desde la unidad de programa que esta siendo probada. Se examina las

estructuras de datos locales para asegurar que los datos mantienen la integridad durante todos los pasos de ejecución del algoritmo. Se prueban las condiciones límites para asegurar que el módulo funciona correctamente con los límites establecidos. Se ejercitan todos los caminos independientes para asegurar que todas las sentencias de código se ejecutan al menos una vez. Y por ultimo se prueban todos los caminos de manejo de errores.

Esta prueba es importante aplicarla a cualquier software en desarrollo porque los casos de pruebas detectan errores esenciales durante el desarrollo del código tales como:

- Comparaciones entre tipos de datos distintos.
- Operaciones de procedencia incorrecta.
- Igualdad esperada cuando los errores de precisión la hacen poco probable.
- Variables o comparaciones incorrectas.
- Terminación de bucles inapropiada o inexistente.
- Variable de bucles modificados de forma inapropiada.
- Interrupción de la ejecución del programa por entradas no válidas de datos.
- La carencia de una ayuda del sistema que pueda guiar al usuario en su utilización.
- En el eliminar de cada operación de las peticiones de cambio.
- En los eventos anidados
- En el funcionamiento de la interfaz del sistema.

4. Recursos

Para la realización de las pruebas fue necesario constar algunos recursos tangibles, dentro de los cuales se pueden mencionar los siguientes:

Como mínimo una PC que como requerimiento de software debe tener:

- Windows 95 o superior

- Office 97 o superior
- Java(Eclipse)
- SQL Server
- Instalar la aplicación (Release)

Y como requerimiento de hardware:

- Microprocesador Pentium o superior
- Al menos 64 MByte de memoria RAM
- Tarjeta de Red
- Mínimo de 20 GByte de Disco duro

Las pruebas realizadas a cada modulo se les realizaron desde una pc que se encontraba en una red local con otros terminales. Las pruebas se diseñaron y se probaron por un probador y es por esto que el tiempo estimado de duración por cada módulo a probar sea de 1 mes.

2.4 Arquitectura de automatización de prueba

La arquitectura de automatización de prueba proporciona una descripción arquitectónica comprensiva del sistema, para ello usa diversas visiones para representar arquitectónicamente aspectos de sistema. Proporciona características fundamentales de la automatización de la prueba del software. Permite observar aspectos claves del sistema tales como lo son la capacidad de mantenimiento, la extensibilidad, confiabilidad, concurrencia, distribución, seguridad y recuperación.

La arquitectura de automatización de prueba se debe llevar a cabo al final de la fase de inicio y elaboración, permitiendo probar y evaluar los diferentes artefactos del sistema.

Existen diversas arquitecturas de automatización de prueba, que muchas se pueden adaptar al software que se esta desarrollando y otras no. Cada proyecto requiere de diferentes variaciones de técnicas y herramientas que afectan la arquitectura de automatización de prueba. La adaptación de una arquitectura se

decide en la fase de elaboración y se extenderá a cada una de las iteraciones de esta fase, y se extiende también a la fase de construcción y de transición.

La automatización de la prueba proporciona algunos beneficios importantes para el desarrollo del software/sistema tales como:

- El aumento en la productividad.
- Reducción de las pruebas manuales y de los errores humanos.
- Posibilita una mayor cobertura de prueba en menor tiempo.
- Reducción de tiempo en la liberación de la aplicación.
- Disminuye los costos y mejorara la calidad del software.

La mayoría de las herramientas que se utilizan para la automatización de la prueba se centran en una actividad o en un grupo específicas de actividades.

Cuando se evalúan las herramientas de automatización de prueba, es importante conocer las limitaciones de las herramientas que se utilizan, y las actividades que automatiza. Las herramientas de prueba se evalúan y se adquieren a menudo basándose en los métodos de:

- Caja blanca
- Especialización de la función
- Caja negra

Las herramientas de la prueba de función se pueden categorizar por las funciones que se realizan. Las designaciones típicas de la función para las herramientas incluyen:

Herramientas para la adquisición de datos que adquieren los datos que se utilizarán en las actividades de la prueba. Los datos se pueden adquirir con la extracción, la transformación, o la captura de datos existentes, o con la generación de casos del uso.

Herramientas estáticas que analizan la información que se contiene en los modelos de diseño, en el código fuente, u otras fuentes fijas. En la información de las métrica

de la calidad tal como complejidad, capacidad de mantenimiento, o líneas del código.

Herramientas dinámicas de la medida que realizan un análisis durante la ejecución del código. Las medidas incluyen la operación run-time del código tal como memoria, detección de error, y funcionamiento.

Una herramienta que se utiliza para la realización de las prueba y que es muy eficaz por las diferentes ventajas que aporta para el desarrollo de la prueba es JUNIT, es una herramienta de automatización de prueba a considerar para el desarrollo de cualquier software desarrollado sobre la plataforma Java, y es por este motivo que se propone usarla en el resto de los proyectos, esta herramienta se esta usando en el SIGIA, porque permite:

- Escribir el código más deprisa a la vez que incrementa la calidad.
- JUnit es elegante y simple.
- Permite que las pruebas comprueben el resultado y proporciona un diagnóstico inmediato.
- Permite componer las pruebas en una jerarquía de conjuntos de pruebas.
- Las pruebas con JUnit no es costoso, es gratis.
- Permite incrementar la estabilidad de los programas.
- JUnit permite que los que hacen las pruebas trabajen como los que hacen el programa.
- Permite escribir pruebas en Java.

JUnit es un conjunto de librerías que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java. Es un conjunto de clases (framework) que permite realizar la ejecución de las clases en Java de manera controlada. Permite evaluar el funcionamiento de cada uno de los métodos de la clase, verificando si esta lo hace de forma correcta. Se evalúa algún valor de entrada y se mira el valor de retorno esperado, si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba, en caso de que

el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

JUnit es un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

En la actualidad las herramientas de desarrollo como NetBeans y Eclipse cuentan con plug-ins que permiten que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se realice de manera automática, facilitando al programador enfocarse en la prueba y el resultado esperado, y dejando a la herramienta la creación de las clases que permiten coordinar las pruebas.

2.5 Datos de prueba

Los datos de prueba del SIGIA cubren los posibles valores de cada parámetro basado en los requerimientos de la base de datos. El caso de prueba se ejecuta una vez por cada combinación de valores. Se escoge 3 valores de cada clase de equivalencia (conjunto de valores que deberán ser tratados iguales).

Ejemplos de clases de equivalencia:

Cadenas

- cadena vacía
- cadena consistente de únicamente un espacio vacío
- sintácticamente correcta: valores cortos y largos
- sintácticamente correcta: valores semánticamente válidos e inválidos
- valor sintácticamente incorrecto: caracteres o combinaciones ilegales
- Para probar caracteres especiales como #, ", ', &, y <

Números

- cadena vacía, si es posible

- 0
- pequeños y largos en rangos positivos
- pequeños y largos en rangos negativos
- fuera del rango de positivos
- fuera del rango de negativos
- que comiencen con ceros
- sintácticamente inválidos (por ejemplo, que incluya letras)
- Fuera del rango de valores permitidos

Identificadores

- cadena vacía
- valor sintácticamente correcto
- sintácticamente correcto: referencia a un ID existente, referencia inválida
- valor sintácticamente incorrecto

Juegos de Datos del Módulo nomencladores de Caja Negra

Campo de la interfaz	Datos Válidos	Datos Inválidos
Código	45,00,99,01,25(Números de dos lugares)	Letras(No admite), Vació, 1€.4&.
Nombre	Cualquier combinación de letras(Carne,Pollo,bebidas,Carnicería),hasta 30 caracteres	Números(001999,999999)
Descripción	Cualquier comentario del nuevo grupo	Ningún dato inválido
Activo	Activo y de insumo	Ningún dato inválido

Útil	Útil y activo	Ningún dato inválido
Insumo	Útil y activo	Ningún dato inválido
Crear	Crear	Ningún dato inválido
Guardar	Todos los campos estén correctos	Que el código no esté correctamente
Métrica	Real o Entero	Ningún dato inválido
Grupo	Seleccionar un grupo	No seleccionar un grupo
Familia	Seleccionar una familia	No seleccionar una familia
Código	0000,1111,1235,9999(4 caracteres)	456897(números mayor de 4 cifras).Letras (qerw) Caracteres especiales como (#, ", ', &, y <)
Unidad/Medida Base	Seleccionarla	Ninguna
Tipo de producto	Seleccionar	Ninguna
Activo	Activado	Ninguna
Controla Existente	activado	Ninguna
Tipo de producto	Letras(Cualquier combinación de letras)	Números(Permitir cadenas de caracteres como (#, ", ', &, y <)), Cadenas vacías

Juegos de Datos del Modulo Administración de Caja Negra

Campo de la interfaz	Datos Validos	Datos Inválidos
Carné de Identidad	CI = 11 caracteres	11<CI<11
contraseña	Contraseña > =5	Contraseña < =5

2.6 Casos de pruebas de Caja Negra de los módulos Nomencladores y Administración

Para la realización de las pruebas de caja negra y para obtener los casos de prueba en este método se utiliza la técnica de las clases de equivalencia, la cual es muy efectiva a la hora de probar la validez de cada uno de los datos que se introducen en el sistema a través de sus entradas.

Con esta prueba se comprobó la funcionalidad y validez de todas las entradas de datos en cada módulo, así como también se determinó si el sistema consta de una interfaz amigable para los usuarios.

Pruebas funcionales del sistema e interfaz

Otros aspectos a medir en la Prueba de la Caja Negra son la existencia y funcionalidad de todos los requerimientos funcionales definidos en la etapa de análisis por los analistas, que según el cliente debía cumplir el sistema, además de la existencia o no de errores en la interfaz del producto.

En la documentación elaborada por los analistas para el análisis del sistema se incluyeron un número de responsabilidades o requerimientos que debía cumplir el sistema para lograr los objetivos propuestos durante la etapa de desarrollo del Sistema de Gestión de Inventario de Almacenes.

Estos requisitos (Ver Anexo #2) son los que debe cumplir el sistema, los cuales fueron agrupados para los dos módulos que están implicados en la prueba.

Pruebas de Caja Negra al módulo nomencladores

En este módulo se probaron los siguientes casos de usos:

1. Nomenclar Grupo Familia
2. Nomenclar unidad de medidas
3. Nomenclar Producto
4. Especialidad Proveedor
5. Tipo de producto
6. Nomenclar Clasificación de Sección
7. Tipo de sección
8. Nomenclar Clientes Terceros
9. Tipo de proveedor

Caso de prueba (CPR) del caso de uso Nomenclar Grupo Familia

Este caso de uso tiene como propósito permitir al usuario crear un nuevo grupo de familia, permitiéndolo primero crear el grupo con todas sus características y después crear las familias asociadas a ese grupo, también permite eliminar cualquier grupo existente y dentro de este cualquier familia perteneciente a este grupo, en la eliminación se le da la opción al usuario de tener activado o no al grupo de familia.

Las pruebas realizadas a este Caso de Uso (Ver Anexo #3) son las siguientes:

- Crear un nuevo grupo al sistema.
- Eliminar un grupo ya creado.
- Crear una familia dentro de un grupo existente.
- Eliminar una familia ya creada.

1 CPR 1.1: Registrar un nuevo grupo al sistema.

Descripción

Este caso de uso brinda la posibilidad de crear un nuevo grupo.

Flujo Central

1 El sistema muestra la interfaz de la aplicación. El usuario accede al menú ***Definiciones*** y escoge la opción “Grupo Familia”, dentro de este la opción grupo de familia.

2 El sistema muestra una página donde ofrece un listado de los grupos ya existentes en la base de datos por “nombre” y “estado”, además de un vínculo para agregar un “nuevo grupo”.

3 El usuario selecciona la opción “Crear”.

4 El sistema muestra al usuario una nueva ventana con campos específicos para crear un grupo, además muestra un vínculo para las familias

5 El usuario llena los datos del grupo a crear, y en caso de no querer nomenclar familias, pincha en el botón “Guardar” para crear el grupo con los datos especificados.

6 El sistema inserta en la base de datos el nuevo grupo registrado y muestra al usuario la pagina anterior donde mantiene actualizados los grupos existentes de la base de datos con el “nombre” y “estado” pertenecientes a cada grupo.

Condiciones de ejecución.

El usuario debe estar autenticado.

CPR 1.2: Eliminar un grupo del sistema.

Descripción

Este caso de uso brinda la posibilidad de eliminar un grupo ya creado en el sistema.

Flujo Central

1. El sistema muestra la interfaz de la aplicación. El usuario accede al menú ***Definiciones*** y escoge la opción “Grupo Familia”, dentro de este la opción grupo de familia.

2. El sistema muestra una página donde ofrece un listado de los grupos ya existentes en la base de datos por “nombre” y “estado”, además de un vínculo para agregar un “nuevo grupo”.
3. El usuario accede al listado del grupo y escoge el grupo que desea eliminar dentro del listado ofrecido.
4. El sistema muestra al usuario la ventana de edición de un grupo con los datos con los que se creó este grupo.
5. El usuario si este grupo esta activado lo desactiva y pincha en el botón “Guardar” guarda el cambio y en caso de no querer eliminar ninguna familias, pincha en el botón “Guardar” para eliminar el grupo anteriormente desactivado

Condiciones de ejecución.

El usuario debe estar autenticado.

Debe haber al menos creado un grupo para poder eliminar algún dato.

CPR 1.3: Crear familia dentro de un grupo existente.

Descripción

Este caso de uso brinda la posibilidad de crear familias asociadas a los grupos ya existentes.

Flujo Central

1. El usuario selecciona un vínculo de los grupos ya listados, al cual se le va a asociar una familia.
2. El sistema muestra al usuario una nueva ventana de edición de grupo donde tiene un botón “Familias”.
3. El usuario accede a través de este vínculo a la página nomenclar familias donde aparece un vínculo de edición de familias “crear familia”, y un listado de las familias asociadas hasta el momento por “nombre” y “estado”.
4. El usuario llena los datos de la familia a crear y pincha en el botón “Crear” para crear la familia con los datos especificados.

5. El sistema inserta en la base de datos la nueva familia asociada y muestra al usuario una página con el listado de las familias donde están listadas las familias existentes de la base de datos con el “nombre” y “estado” pertenecientes a cada una.

CPR 1.4: Eliminar una familia existente dentro de un grupo creado.

Descripción

El caso de uso permite eliminar una familia dentro del grupo asociado

Flujo Central

1. El sistema muestra una página donde ofrece un listado de las familias ya existentes en la base de datos por “nombre” y “estado”.
2. El usuario selecciona un vínculo de las familias ya listados,
3. El usuario desactiva la familia que desea eliminar.
4. El usuario pincha en el botón “Guardar“ y guarda el cambio
5. El usuario vuelve a marcar la anterior familia seleccionada y pincha en el botón “Eliminar”
6. El sistema elimina correctamente la familia asociada al grupo anterior

Condiciones de ejecución.

Debe haber al menos una familia creada para poder eliminar algún dato.

Caso de prueba del caso de uso nomenclar Unidad de Medida

Este caso de uso permite crear, y eliminar las unidades de medida producto (U/M) con las que trabajará la entidad.

Las pruebas realizadas a este Caso de Uso (Ver Anexo #4) son las siguientes:

- Crear unidad de medida.
- Eliminar Unidad de Medida

2 CPR 2.1: Crear unidad de medida.

Descripción

Este caso de prueba brinda la posibilidad de crear una unidad de medida.

Flujo Básico

1. El usuario selecciona del menú Definiciones la opción nomenclar Unidad de Medida.

2. El sistema despliega la página para nomenclar U/M, donde aparece un botón para crear una nueva unidad de medida

3. Si el usuario decide:

a) Crear una nueva U/M va a la acción 3.2.1

b) Modificar una U/M va a la acción 3.2.2.

Flujo Central

1. El usuario presiona el botón “Crear” que aparece en la página principal
2. El sistema muestra la página Editar para crear U/M que tiene todos los elementos de la U/M que serán introducidos. Además tiene un botón “Guardar” y “Cancelar”.
3. El personal de economía introduce los datos de la U/M:

- Nombre
- Métrica (real o entero)
- Descripción.
- Activo

3.1 Presiona el botón Crear para guardar los datos el sistema se mantiene en la página Editar, y limpia todos los campos.

Flujo alterno

2. Si el personal de economía, presiona el botón “Cancelar” retorna a la página principal.

Condiciones de Ejecución

Todos los campos deben estar correctamente introducidos.

CPR 2.2: Eliminar Unidad de Medida

Descripción

Este caso de prueba brinda la posibilidad de eliminar una unidad de medida.

Flujo Central

1. El sistema muestra la página “Editar” donde aparecen todos los campos activos que pueden ser eliminados.

3. El personal de economía selecciona la unidad de medida a eliminar.

3.1 El personal de economía deshabilita la unidad de medida si esta activada.

4. El personal de economía presiona el botón “Guardar” y el sistema guarda correctamente los campos

5. Selecciona nuevamente la unidad desactivada y pincha el botón “Eliminar” y elimina correctamente la unidad de medidas.

Condiciones de Ejecución

Cuando se desee desactivar una U/M, el personal de economía debe desmarcar el campo Activo Una vez desmarcado dicho campo, no se podrá usar más esta U/M hasta que no esta activada otra vez.

Caso de prueba del caso de uso nomenclar Especialidad Proveedor

El presente caso de uso tiene como objetivo posibilitar la creación de las especialidades con las que se asociaran posteriormente los distintos proveedores que surten la entidad. Mediante el presente flujo de eventos además de registrar una nueva especialidad se puede buscar el listado de los proveedores.

Las pruebas realizadas a este Caso de Uso (Ver Anexo #5) son las siguientes:

- Crear Especialidad de Proveedores.
- Eliminar una Especialidad de Proveedores.

3 CPR 3.1: Crear Especialidad de Proveedores.

Descripción

Este caso de prueba brinda la posibilidad de crear una nueva especialidad proveedor con sus características.

Flujo Central

1. El jefe de compras selecciona de la parte superior de la página el menú Definiciones la opción “Especialidad de proveedor”.
2. El sistema despliega una interfaz donde se muestran las especialidades antes registradas atendiendo a dos aspectos y te permite volver a llenar ,los campos:
   - Código
   - Nombre
   - Descripción

   Asociar Familias al grupo
   - Enlazar familias disponibles y Familias seleccionadas
   - El usuario pulsa el botón “Crear”

3. El actor no desea crear ya la nueva especialidad proveedor solo presiona el botón “Cancelar”.Y el sistema limpia los campos para inicial la operación anterior.

Condiciones de Ejecución

El usuario debe estar autenticado en el sistema.

CPR 3.2: Eliminar Especialidad Proveedor

1. El jefe de compras selecciona de la parte superior de la página el menú Definiciones la opción “Especialidad de proveedor”.
2. El sistema despliega una interfaz donde se muestran las especialidades antes registradas atendiendo a dos aspectos:
   - Nombre
   - Estado

- Asociar familias
- Grupo y familias Disponibles

3. El jefe de compras pincha el botón Deshabilita la especialidad si está activada
4. El jefe de compras guarda los cambios anteriormente realizados.
5. Es jefe de compras escoge la especialidad que desea eliminar y pincha el botón “Eliminar“ el sistema elimina la especialidad.

Condiciones de Ejecución

El sistema debe tener ya registrada alguna especialidad proveedor.
El usuario debe estar autenticado en el sistema.

Caso de prueba del caso de uso nomenclar Producto

Este caso de uso posibilita al Personal de Economía registrar, y eliminar un producto teniendo en cuenta las restricciones correspondientes.

Las pruebas realizadas a este Caso de Uso (Ver Anexo #6) son las siguientes:

- Crear un nuevo Producto
- Eliminar un Producto.

4 CPR 4.1: Crear un producto

Descripción

Este caso de prueba brinda la posibilidad de crear un Producto.

Flujo Básico

1. El personal de economía selecciona del menú general Definiciones la opción nomenclar producto. El sistema muestra la página para crear un nuevo producto.
2. El Personal de Economía hace clic en el botón “Crear”.
3. Se inicia el CU Registrar equivalencias entre UM/producto.

Flujo Central

1. El Personal de Economía hace clic en el botón “Crear”.
2. El sistema muestra la página que contiene los siguientes campos:

- Grupo

- Familia
- Código
- Nombre
- Descripción
- UM/base
- Estado (Activo o no)
- Tipo de producto
- U/M

Y tres botones: “Crear”, “Cancelar”, “Eliminar”.

3. El Personal de Economía introduce los datos y hace clic en el botón “Crear”.

4. El sistema verifica que los datos estén correctos, no puede haber ningún campo vacío y se muestra la misma página Editar con los campos vacíos por si el Personal de Economía quiere registrar otro producto.

Flujo alterno

1. Si El Personal de Economía no desea seguir realizando la acción hace clic en el botón “Cancelar” y se muestra la página que contiene el botón “Crear”.
2. Si alguno de los campos no es correcto el sistema muestra un mensaje de error “Verifique los campos”

Condiciones de Ejecución

El usuario que realice esta operación debe estar registrado y tener la autorización para acceder a esta parte del sistema.

Todos los campos deben estar correctamente introducidos.

CPR 4.2: Eliminar un Producto.

Descripción

Este caso de prueba brinda la posibilidad de eliminar un Producto.

Flujo Central

1. El Personal de Economía selecciona del menú general Definiciones la opción Nomenclar Producto.

El sistema muestra la página para crear un nuevo producto.

3. El Personal de Economía selecciona el producto que desea eliminar

4. Si el producto esta activo lo desactiva y pincha en el botón" Guardar "

5. Selecciona el producto que desea eliminar.

6. El Personal de Economía pincha en el botón "Eliminar".

7. El producto es eliminado correctamente.

Condiciones de Ejecución

Debe de estar registrado en el sistema y tener la autorización para acceder a esta parte del sistema.

El producto a eliminar ya debe estar registrado.

Caso de prueba del caso de uso nomenclar Clientes Terceros

El propósito de este caso de uso es nomenclar un cliente al que se le hará una venta, especificar que el propósito del almacén no es hacer ventas, esto se hará solo en casos excepcionales, con un tipo de producto que tenga lento movimiento o que este en el almacén hace mucho tiempo y que no se le haya dado uso.

Las pruebas realizadas a este Caso de Uso (Ver Anexo #7) son las siguientes:

- Registrar (Crear) un nuevo cliente tercero.
- Eliminar un cliente Tercero.

5. CPR 5.1: Registrar (Crear) un nuevo cliente tercero.

Descripción

Este caso de prueba permite al cliente crear o insertar un nuevo cliente tercero al cual se le hará una venta.

Flujo Central

1. El cliente selecciona de la parte superior de la página el menú Definiciones la

opción “Clientes Terceros”.

2. El sistema despliega una interfaz donde se muestran los campos que se le deben llenar al cliente a registrar, mostrando ,los siguientes campos:
   - Código
   - Nombre
   - Dirección
   - E-mail

CPR 5.2: Eliminar un cliente tercero

Descripción:

El caso de uso brinda la posibilidad de eliminar un cliente tercero de la lista que se visualiza.

Flujo Central

1. El sistema despliega una interfaz donde se muestran los campos que han llenado del nuevo cliente.
2. Si el cliente está Activo, el usuario desactivará el cliente y presiona el botón “Guardar” para guardar los cambios.
3. El usuario marca al cliente que desea eliminar y presiona el botón “Eliminar”.
4. El sistema elimina correctamente al cliente.

Condiciones de Ejecución

Este insertado al menos un cliente tercero para efectuar la actividad.

Caso de prueba del caso de uso nomenclar Clasificación de Inventario

El sistema deberá ser capaz de ofrecerle al usuario la manera de definir las características de las clasificaciones de sección (central de insumo, de mantenimiento, de comestibles, de elaboración, carnicería, etc.), en dependencia del tipo de productos que se manejan en la misma, dándole la posibilidad de que entre estos datos y que estos se guarden en la base de datos. Además deberá permitirle hacer modificaciones en las clasificaciones ya existentes en caso de que lo desee, a todos sus campos exceptuando el código, así como deshabilitarla denegando así su uso.

Las pruebas realizadas a este Caso de Uso (Ver Anexo #8) son las siguientes:

- Editar tipo de Sección.
- Listar tipo de Sección.

CPR 5.3: Registrar nueva clasificación de secciones.

Descripción:

Permite al usuario crear nuevos tipos de secciones introduciendo datos, permite guardarlos y eliminarlos.

Flujo Central

1. El usuario elige de la opción Definiciones nomenclar "Clasificación de sección".
2. Es sistema muestra una interfaz en la que el usuario puede editar todas las propiedades de la nueva clasificación:
   - Nombre
   - Descripción de la nueva sección
   - Tipo (Almacén Central, Sección de Gasto, Sección de elaboración)
   - Activo
   - Si tiene arancel incluido
3. El usuario entra los datos para Restringir familias de Entrada
   - Grupos ya existentes anteriormente creados
   - Familias (Enlazar para mostrar la lista en la parte derecha.)
   - Muestra el código que se le asigna a la sección.

   3 .1 Restringir Familias de Entradas
   - Grupos ya existentes
   - Familias
4. El usuario presiona el botón "Crear".

5. El sistema salva la información en la base de datos y pincha en el botón visualizar para observar la inserción.
6. El usuario presiona el botón “Cancelar” para limpiar los campos y volver hacer lo anterior.
7. El usuario para eliminar una clasificación selecciona las clasificaciones ya definidas y el sistema muestra sus respectivos campos, después el usuario debe desactivar la sección y guarda los cambios y posteriormente pinchar en el botón Eliminar.

Condiciones de Ejecución

Esté registrada la lista de los Grupo/Familia.

Que haya alguna clasificación ya registrada en caso de que se desee hacer una modificación o eliminar.

CPR 5.4: Listar clasificación de sección

Descripción:

En este caso de uso se va a mostrar las informaciones que el usuario desee y necesite de las secciones ya existentes. Le permite hacer búsqueda ya sea por código, nombre, activo, arancel incluido y descripción.

Flujo Central

1. El usuario selecciona de la interfaz listar “Clasificaciones de Sección”.
2. El sistema muestra el listado de todos los campos con la información de las secciones registradas hasta el momento.
3. El sistema te permite hacer búsquedas de las secciones ya existentes ya sea por el nombre, descripción, código, activo, arancel incluido.

Casos de prueba de caja negra al caso de uso Sección de Inventario

El sistema debe permitir al usuario entrar toda la información referente a una nueva sección de inventario y guardarla en la base de datos. Además deberá permitir eliminar una sección ya existente en caso de que el usuario lo desee, así como desactivarla porque de esta manera esta negando su uso.

Las pruebas realizadas a este Caso de Uso (Ver Anexo #9) son las siguientes:

- Registrar sección de inventario.
- Eliminar sección de inventario.
- Verificar que no tenga existencia de productos.

6 CPR 6.1: Registrar sección de inventario

Descripción:

Este caso de prueba brinda la posibilidad de registrar una nueva sección de inventario

Flujo Central:

1. El personal de economía accede a la página principal y elige la opción Definición.
2. El personal de economía da clic en la opción sección de inventario.
3. El sistema muestra una nueva interfaz con un listado con el nombre y el estado de las secciones que ya están activas, y un botón “Crear” para realizar una nueva sección.
4. El usuario presiona el botón “Crear" que se muestra en la interfaz.
5. El sistema muestra una interfaz donde el usuario puede editar todas las propiedades de la sección y además podrá conocer el código que le asigna el sistema a la misma.
6. El personal de Economía entra al sistema los siguientes datos:
   - Nombre
   - Descripción
   - Responsable
   - Dirección
   - Referenciar documentos.
   - MVI.
   - Empresa.
   - Centro de Costo.

- Valoración de inventario.
- Tipo de sección.
- Entidad.
- Valorar existencia
- Activo.

7. El personal de economía presiona el botón “Guardar”.
8. El sistema guarda la información en la base de datos.

Condiciones de Ejecución

Que estén definidos previamente los tipos de sección.

Que el personal de economía este autenticado.

CPR 6.2: Modificar Sección

Descripción:

Este caso de prueba brinda la posibilidad de modificar la información de una sección ya existente, exceptuando el nombre y el tipo de sección.

Flujo Central:

1. El usuario accede a la página principal y elige la opción Definición.
2. El personal de economía elige la opción “sección inventario” de la lista de nomencladores.
3. El sistema muestra una interfaz con las diferentes secciones que ya se han definido previamente, también en la misma interfaz muestra una lista desplegable que se llama “Campo” y te da la posibilidad de poner el “Valor” de ese campo. Aparece un botón “Buscar” para realizar la búsqueda.
4. El personal de economía puede hacer clic en el vínculo de una de las secciones o simplemente poner los datos y dar clic en el botón “Buscar”.
5. El sistema en caso que sea la primera opción muestra una interfaz con la información referente a esa sección que elegiste para modificar.
6. El sistema en caso que sea la segunda opción te muestra una interfaz con la información que elegiste para modificar.

7. El personal de economía en caso que sea la primera opción modifica los campos que desee y luego salva la nueva información.
8. El sistema guarda la información en la base de datos.
9. El personal de economía en caso de que sea la segunda opción modifica la información y luego guarda.
10. El sistema guarda los datos modificados.

Condiciones de Ejecución

Tiene que existir la sección que se quiere modificar

Caso de prueba del caso de uso nomenclar proveedor

Este caso de uso tiene como propósito registrar toda la información de los proveedores que surten a la entidad hotelera. El mismo se inicia cuando el Jefe de Compras selecciona dentro del menú Definiciones la el nomenclador proveedor, una vez allí lleva a cabo la opción deseada: inserta o modifica la información en la interfaz pertinente.

Las pruebas realizadas a este Caso de Uso (Ver Anexo #10) son las siguientes:

- Registrar la información de un proveedor.
- Modificar la información de un proveedor.
- Eliminar un proveedor

7 CPR 7.1: Registrar la información de un proveedor.

Descripción:

Este caso de uso realiza la acción de registrar, insertar un nuevo proveedor a la aplicación

Flujo Central:

1. El usuario selecciona del menú Definiciones la opción “Proveedor”.
2. El sistema despliega una interfaz donde se muestran los campos necesarios para registrar un proveedor:
   - Código.

- Nombre
- Dirección.
- E-mail
- Teléfono.
- Activo
- Especialidades Asociadas.

3. En la parte derecha de la interfaz se muestra un panel donde se encuentran los proveedores antes registrados.
4. El usuario para crear un nuevo proveedor pincha en el Botón. “Crear”
5. Tiene un Chekboo que va a activar al proveedor o lo va a desactivar
6. El usuario pincha en el botón “Guardar”.
7. Guarda la información en la base de datos
8. El usuario pincha el botón “Cancelar” se redirige a la página con los proveedores ya registrados hasta el momento.

Condiciones de Ejecución

Se debe haber nomenclado previamente las especialidades para asociarlas a cada proveedor.

Se debe haber nomenclado previamente las monedas con las que la entidad opera para asociarlas a cada cuenta con la que operará el proveedor.

Se deben haber seleccionado del sistema de contabilidad las cuentas contables para asociarlas a cada proveedor con las que se trabajará en el sistema de inventarios, en este caso de uso se usan del tipo Cuentas por Pagar.

El usuario debe tener los permisos adecuados para realizar el caso de uso.

CPR 7.2: Modificar la información de un proveedor

Descripción:

Este caso de uso modifica la información de un proveedor ya existente en la base de datos

Flujo Central

1. El usuario selecciona del menú Definiciones la opción “Proveedor”.
2. El sistema despliega una interfaz donde se muestran los proveedores antes registrados atendiendo a los factores:
   - Campo
   - Nombre
   - Código
   - Valor
   - Un botón buscar
   - En la misma interfaz aparecen también los campos para mostrar los proveedores ya registrados tales como.
   - Nombre
   - Nacionalidad
   - Estado
3. El sistema muestra una interfaz con los datos que se pueden modificar para hacer cambio, tales como.
   - Dirección.
   - Nacionalidad.
   - Teléfonos.
   - E-mail.
   - Especialidades asociadas
4. El usuario pincha en el botón “Guardar”y el sistema guarda la información en la base de datos.
5. El usuario pincha el botón “Cancelar” se redirige a la pagina con los proveedores

ya registrados hasta el momento.

CPR 7.3: Eliminar la información de un proveedor

Descripción

Este caso de prueba brinda la posibilidad de eliminar un proveedor.

Flujo Básico

1. El jefe de compra selecciona del menú Definiciones la opción Proveedor.
2. El sistema despliega la página para registrar un nuevo proveedor, donde aparece un botón para crear un proveedor

Flujo Central

1. El sistema muestra la página “Editar proveedor” donde aparecen todos los campos activos que pueden ser Eliminados.
2. El Jefe de compra selecciona el proveedor a eliminar
3. El jefe de compra deshabilita el proveedor que está activada
4. El jefe de compra presiona el botón “Guardar” y el sistema guarda correctamente los campos
5. Selecciona nuevamente el proveedor desactivado y pincha el botón eliminar y elimina correctamente la unidad de medidas.

Condiciones de Ejecución

Debe de haber al menos un proveedor registrado.

Pruebas de Caja Negra al módulo Administración

En este modulo se probaron los siguientes casos de Uso

Gestionar Usuario

Gestionar Roles

Gestionar Terminales

Caso de prueba del caso de uso Gestionar Terminales

Este caso de uso comprende la funcionalidad de crear estaciones de trabajo desde donde se ejecutara la aplicación.

Las pruebas realizadas a este Caso de Uso (Ver Anexo #11) son las siguientes:

- Crear un nuevo Terminal
- Eliminar un Terminal

CPR 8.1: Crear un nuevo Terminal

Descripción

Este caso de uso permite insertar un nuevo Terminal para que el usuario pueda ejecutar la aplicación desde su puesto laboral

Flujo Central

1. El usuario accede al menú Administración y señala la opción “Terminales”.
2. El sistema le muestra la forma de entrar los datos necesarios
3. El usuario introduce los datos: Nombre de máquina y si desea crearla activa (este campo aparece activo por defecto).
4. El sistema verifica que el nombre de la Terminal no esté registrado en el sistema.
5. El sistema registra la nueva Terminal.
6. El sistema verifica que el nombre de la maquina ya existe en el sistema y muestra el mensaje de error indicando el error.
7. Sino existe crea la nueva Terminal
8. El sistema guarda correctamente los datos en la base de datos.

Condiciones de Ejecución

El sistema debe ser instalado y ejecutado correctamente.

CPR 8.2: Eliminar un Terminal

Descripción

Este caso de uso permite eliminar un Terminal existente o modificarlo para nuevamente guardarlo.

Flujo Central

1. El sistema le muestra un listado con todos los terminales registrados en la base de datos
2. El usuario marca el Terminal que desea eliminar de la lista.
3. El sistema le muestra si esta activo o no.
4. Si el usuario desea modificar algún dato puede hacerlo y luego pinchar en el botón Guardar
5. Para eliminar tiene que cerciorarse que la Terminal no esta activa, y si lo está debe desmarcarla.
6. El usuario debe pinchar en el Botón “Guardar” para guardar los nuevos cambios realizados
7. El usuario selecciona el mismo Terminal desactivado y l pincha en el botón” Eliminar”
8. El sistema elimina correctamente los datos.

Condiciones de Ejecución

Que exista el menos un Terminal registrado.

Caso de prueba del caso de uso Gestionar Usuario

Este caso de uso comprende la funcionalidad de crear usuarios, cambiar su rol de usuario, cambiar el password. Entre otras funcionalidades de entradas

Las pruebas realizadas a este Caso de Uso (Ver Anexo #12) son las siguientes:

- Registrar un usuario
- Eliminar Usuario

9 CPR 9.1: Registrar un usuario

Descripción

Este caso de prueba permite crear un usuario que no este en la base de datos.

Flujo Central

1. El usuario accede a la aplicación y en la parte superior pincha en el menú de Administración y posteriormente en la pestaña de Editar usuario.
2. El usuario introduce los datos:
   - Nombre del usuario
   - Primer apellido
   - Segundo apellido
   - CI
   - Cargo
   - Usuario
   - Password
   - Rol
   - Crear
3. El sistema verifica que el usuario sea único.
4. El sistema verifica que el password tenga como mínimo 8 caracteres. El password se debe guardar de forma encriptado (md5).
5. El sistema verifica que el carné de Identidad tenga 11 caracteres.
6. El usuario presiona el botón “Guardar” y sistema guarda correctamente los datos.

Condiciones de Ejecución

El usuario que vaya a iniciar el caso de uso debe de estar loggeado como Administrador del Sistema.

El caso de prueba “Eliminar un usuario” se hace a la perfección por lo que no se crea un caso de prueba porque no es necesario. Cambiar contraseña se hace satisfactoriamente.

Caso de prueba del caso de uso Gestionar Roles

Este caso de uso tiene la funcionalidad de crear roles de usuarios con las opciones de accesibilidad para cada uno.

Las pruebas realizadas a este Caso de Uso (Ver Anexo #13) son las siguientes:

- Registrar nombre del Rol
- Eliminar Rol

9 CPR 9.1 Registrar nombre del Rol

Descripción

Este caso de prueba tiene la funcionalidad de crear roles para los usuarios insertados

Flujo Central

1. El usuario accede a la pestaña “Crear Rol de Usuario”.
2. El sistema le muestra la forma de entrar los datos necesarios.
3. El usuario introduce los datos: Nombre del Rol, y escoge de la lista de opciones las que le quiera asignar a dicho rol.
4. El sistema verifica que el nombre del rol no esté repetido

Lista de Opciones:

- Administrar sistema
    a. Crear/Editar Usuarios
    b. Eliminar Usuarios
    c. Crear / Editar Roles Usuarios
    d. Eliminar Roles Usuario
    e. Activar / Desactivar Estaciones de Trabajo
    f. Eliminar Estaciones de Trabajo
- Clasificadores
    a. Crear/Editar Clasificación de Sección
    b. Eliminar Clasificación de Sección
    c. Crear/Editar Grupo/Familia de Productos

d. Eliminar Grupo/Familia de Productos
e. Crear/Editar Producto
f. Eliminar Producto
g. Crear / Editar Enlaces G/F/S
h. Eliminar Enlaces G/F/S
i. Crear /Editar Secciones de Inventario
j. Eliminar Secciones de Inventario
k. Crear /Editar Unidades de Medida
l. Eliminar Unidades de Medida
m. Crear / Editar Proveedor
n. Eliminar Proveedor
o. Crear /Editar Especialidad de Proveedor

- El sistema refresca la interfaz mostrando el nuevo rol creado.

Condiciones de Ejecución

El sistema debe ser instalado y ejecutado correctamente.
Se debe cargar del sistema las opciones a las que se podrá tener acceso en el sistema.

El botón “Eliminar” de la interfaz elimina correctamente el rol siempre y cuando no este asociado a ningún usuario. Por lo que como los pasos se ejecutan correctamente no se elaboró un Caso de prueba formal.

2.7 Casos de pruebas de Caja Blanca para el módulo Nomencladores

Para la obtención de los casos de prueba en este método de Caja Blanca se utilizó la técnica del camino básico, la cual es muy efectiva para obtener todas las variantes en las que puede correr un procedimiento.

Con esta prueba se comprueba si el programa en cada una de sus variantes llegó al resultado esperado, si las condicionales toman el valor adecuado en el momento preciso.

Método

Public class Especialidad _ proveedor_ implements Especialidad_Proveedor_Service

Caminos Independientes

Camino1: 1-2-3-4-5-12-13-30-34

Camino2: 1-2-3-4-5-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-34

Camino3: 1-2-3-4-5-12-13-14-15-16-17-19-20-21-22-24-25-26-27-34

Camino4: 1-2-3-4-5-12-13-14-15-16-17-19-20-16-17-18-19-20-21-22-23-34

Camino5: 1-2-3-4-5-12-13-14-15-16-17-18-19-20-16-17-19-20-21-22-24-25-26-27-34

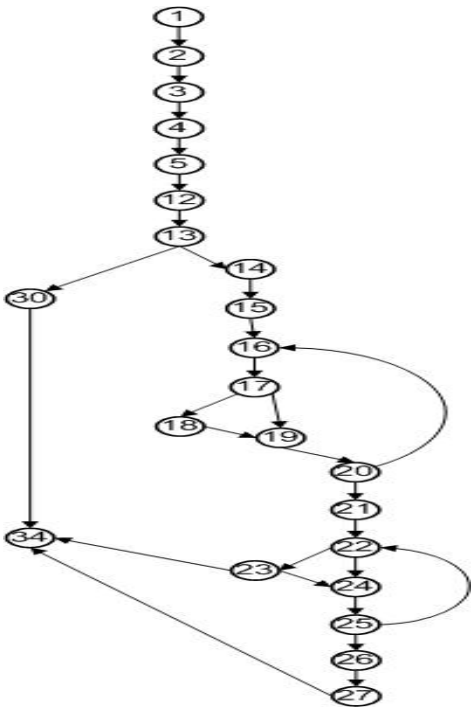
Camino6: 1-2-3-4-5-12-13-14-15-16-17-19-20-21-22-24-25-22-23-34

Camino7: 1-2-3-4-5-12-14-15-16-17-18-19-20--21-22-24-25-22-24-25-26-27-34

Complejidad Ciclomática

V (G) =28-23+2=7

Grafo de Flujo



Casos de pruebas

Caso #1

Public boolean delete Especialidad _ proveedor (AEspecialidadproveedor especialidad)

If (!f especialidad.get Activo())

Return false

Resultado Esperado (R.E) Si no está activo se puede eliminar la especialidad proveedor.

Caso #2

Especialidad.get activo ()==True

R.E Como está activo entra al ciclo y recorre todas las tuplas de la base de datos para eliminar los datos activos para así poder eliminar la especialidad proveedor.

Caso #3

Public boolean delete especialidad proveedor (Aespecialidad_proveedor)

R.E Se recorre la lista de especialidad proveedor para verificar si algún proveedor especialidad proveedor tiene esa especialidad y lo adiciona a listaR.add (proveedor)

Caso #4

R.E Se recorre la lista y se elimina una especicialidad proveedor si se elimina el proveedor.

Nota: los caminos 5, 6, 7 ya forman parte de los casos de pruebas del camino1, 2, 3, 4.

Método

Public boolean delete_Sección (ASeccion seccion)

Caminos Independientes

Camino1:1-2-17-21

Camino2:1-2-3-4-5-6-7-8-9-21

Camino3: 1-2-3-4-5-6-7-8-11-12-13-14-15-21

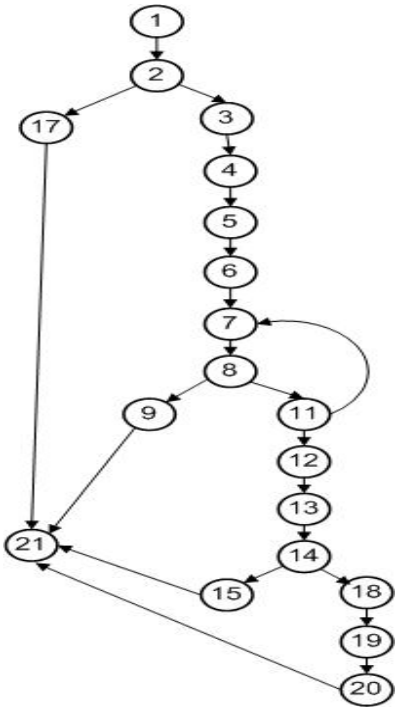
Camino4: 1-2-3-4-5-6-7-8-11-12-13-14-18-19-20-21

Camino5: 1-2-3-4-5-6-7-8-11-7-8-11-12-13-14-18-19-20-21

Complejidad Ciclomática

V (G) =22-19+2 =5

Grafo de Flujo



Casos de pruebas

Camino #1

Método

Public boolean delete_Sección (Aseccion seccion)

Seccion.get activo==True

R.E Si es activa no se puede eliminar la seccion y return falso

Camino #2

Seccion.get activo==False

R.E Si la sección no es activa recorre la lista de las tarjetas de estiva busca todas las tarjetas de estivas y verifica que el saldo sea cero.

Camino #3

Entrada=Seccion

R.E Busca las tarjetas de estiva asociadas a la seccion para comprobar que el saldo de los productos es cero y en caso afirmativo verifica que los documentos primarios asociados a esa sección no esté pendiente para poder eliminar la sección y retorna true.

Camino #4

Entrada=Seccion

R.E Si ocurre un fallo en el proceso de eliminación es capturado en la excepción y retorna falso.

Camino #5 Hace el mismo recorrido que el camino 3.

Método

Public boolean delete Proveedor (AProveedor proveedor)

Caminos Independientes

Camino1:1-2-3-4-5-8-4-5-6-8-9-10-16

Camino2:1-2-12-16

Camino3:1-2-3-4-5-8-9-13-14-16

Camino4:1-2-3-4-5-6-8-16

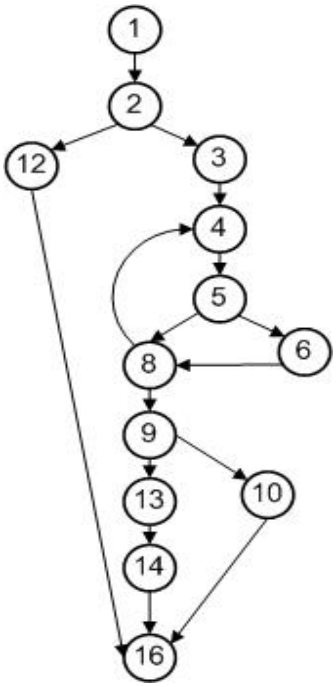
Camino5: 1-2-3-4-5-8-9-10-16

Camino6: 1-2-3-4-5-8-4-5-6-8-9-13-14-16

Complejidad Ciclomática

V (G) =17-13+2=6

Grafo de Flujo



Casos de pruebas

Camino #1

p.get Activo==False

R.E Si no está activa verifica que no existe ningún informe de recepción con estado "pendiente", y que no existe ninguna factura de compra pendiente asociada a ese proveedor.

Camino #2

Public boolean delete_ proveedor (Aproveedor p)

p.get Activo==True

R.E Si está activa no se puede eliminar Return falso.

Camino #3

Entrada =p

R.E Si ocurre un fallo en el proceso de eliminación de un proveedor es capturado por la excepción y retorna falso.

Camino #4

Entrada =p

p.get activo== true

R.E Si el proveedor está activo recorre la lista de informes de recepción y si y si el estado de los informes de recepción es pendiente y existe la factura de compra asociada al informe de recepción el proveedor no se elimina retorna falso.

Camino #5 El camino 5 depende de los caminos ya recorridos.

Camino #6 Es la combinación del 3 con el resto de los caminos.

Método

Public boolean Eliminar_Unidad_Medida (AUnidad_Medida um)

Caminos Independientes

Camino1:1-2-11-15

Camino2:1-2-3-4-5-6-7-8-9-10-15

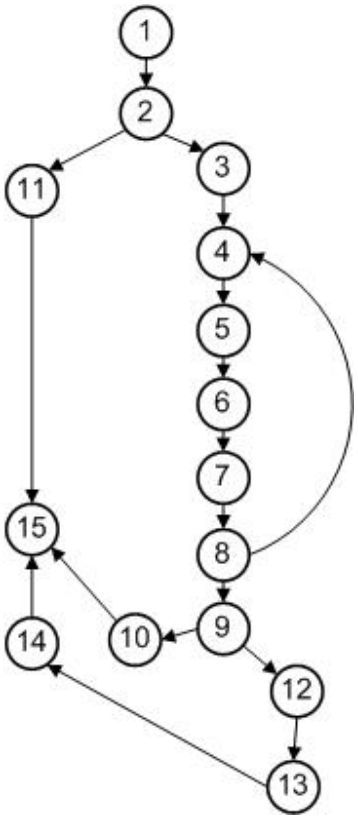
Camino3:1-2-3-4-5-6-7-8-9-12-13-14-15

Camino4: 1-2-3-4-5-6-7-8-4-5-6-7-8-9-12-13-14-15

Complejidad Ciclomática

V (G) =17-15+2=4

Grafo de Flujo



Casos de pruebas

Camino #1

Public boolean Eliminar_Unidad_Medida (AUnidad_Medida um)

um.get_activa==true

R.E Si es activa la unidad de medida no se puede eliminar la seccion y return falso

Camino #2

um.get activo==False

R.E Si la unidad de medida no es activa recorre la lista de productos que tienen asociado una unidad de medida y si alguno está activo no elimina la unidad y retorna falso.

Camino #3

Entrada=um

R.E Si ocurre un fallo en el proceso de eliminación de una unidad de medida es capturado por la excepción y retorna falso.

Camino #4 depende de los caminos ya recorridos.

Método

Public boolean delete_Producto (Aproducto producto)

Caminos Independientes

Camino1:1-2-37-28

Camino2:1-2-3-4-5-6-7-8-9-10-22-23-24-25-26-28

Camino3: 1-2-3-4-5-8-9-10-11-12-15-16-17-20-21-23-24-25-26-28

Camino4: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-17-20-21-23-24-25-26-28

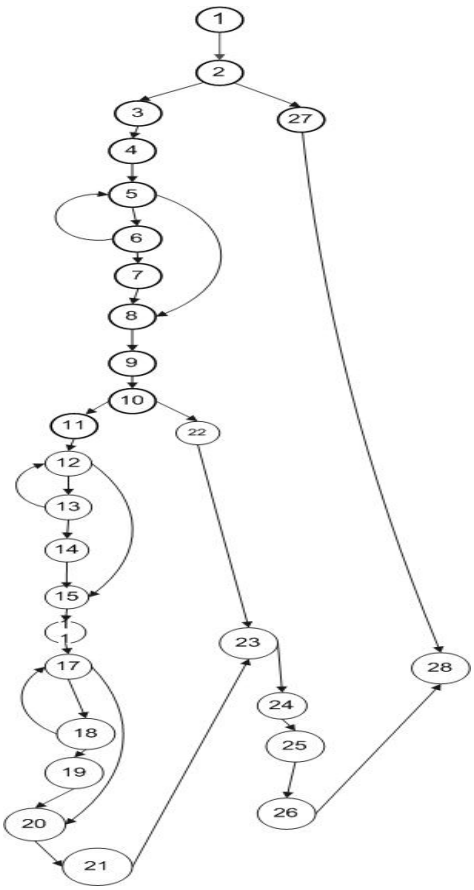
Camino5: 1-2-3-4-5-6-5-8-9-10-11-12-13-14-15-16-17-18-19-20-21-23-24-25-26-28

Camino 6: 1-2-3-4-5-6-5-8-9-10-12-13-12-13-14-15-16-17-18-17-19-20-21-21-23-24-25-26-28

Camino 7: 1-2-3-4-5-6-7-8-9-10-11-12-15-16-17-20-21-23-24-25-26-28

Camino 8: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-17-26-21-23-24-25-26-28

Camino 9: 1-2-3-4-5-6-7-8-9-10-11-12-13-12-15-16-17-20-21-23-24-25-26-28



Complejidad Ciclomática

V (G) =35-28+2=9

Grafo de flujo

Casos de pruebas

Camino #1

Public boolean delete_Producto(AProducto producto)

Producto.get_activo == true

R.E Si el producto está activo no se puede eliminar el producto y retorna falso.

Camino #2

Entrada= producto

R.E Se verifica que el producto que se pasa como parámetro no esté activo, se crea una lista de BaseObject llamada list PTEG, se crea otra lista de BaseObject denominada list y se inicializa con la lista de tarjetas de estiba general de cada producto, luego se recorre todas las listas de tarjetas y en caso de que el Código del producto de algunas de las tarjetas es igual al código del producto pasado por parámetros se agrega a la lista PTGE y se comprueba que el saldo de todas las tarjetas sea distinto de cero y se retorna falso.

Camino #3

Entrada= producto

R.E Se recorre todas las listas de tarjetas y en caso de que el Código del producto y el de la tarjeta sean iguales se agrega a la lista PTGE y se comprueba que el saldo de las tarjetas sea igual cero se elimina el producto entrado y retorna true.

Camino #4

Entrada= producto

R.E Si ocurre un fallo en el proceso de eliminación de un producto es capturado por la excepción y retorna falso.

Nota: los caminos 5, 6, 7, 8,9 están implícitos en el recorrido del resto de los caminos ya explicados.

2.8 Logs de Pruebas

El histórico de pruebas (test log) documenta todos los hechos relevantes ocurridos durante la ejecución de las pruebas

Identificador:

- Descripción de la prueba: elementos probados y entorno de la prueba
- Anotación de datos sobre cada hecho ocurrido (incluido el comienzo
- y el final de la prueba)
- Fecha
- Cantidad de errores.
- Otras informaciones

Logs de las pruebas de Caja Negra del módulo nomencladores

Responsable	Descripción de la Prueba. Elementos Probados	Entorno de prueba	Anotación sobre cada hecho ocurrido en cuanto a : Fecha	Cantidad de errores
Nayrobi Fuentes	Caso de Uso probado Especialidad proveedor. Se aplicaron dos pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	17/4/2007	5
Nayrobi Fuentes	Caso de Uso probado Proveedor. Se aplicaron tres pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	3/5/2007	9

Nayrobi Fuentes	Caso de Uso probado Sección de Inventario. Se aplicaron tres pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	2/05/2007	6
Nayrobi Fuentes	Caso de Uso probado Producto. Se aplicaron dos pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	6/04/2007	5
Nayrobi Fuentes	Caso de Uso probado Clasificación de Sección. Se aplicaron dos pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	15/04/07	3
Nayrobi Fuentes	Caso de Uso probado Unidad de Medida. Se aplicaron dos pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	26-04-2007	9
Nayrobi Fuentes	Caso de Uso probado Clientes Terceros. Se aplicaron dos pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	25 /4/2007	4

Nayrobi Fuentes	Caso de Uso probado Grupo Familia Se aplicaron dos pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	19-/04/2007	13

Logs de las pruebas de Caja Negra del módulo Administración

Responsable	Descripción de la Prueba. Elementos Probados	Entorno de prueba	Anotación sobre cada hecho ocurrido en cuanto a : Fecha	Cantidad de errores
Nayrobi Fuentes	Caso de Uso probado Crear un nuevo Terminal. Se aplicaron dos pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	20/4/20 07	6
Nayrobi Fuentes	Caso de Uso probado Registrar un usuario. Se aplicaron dos pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	4/5/2007	2
Nayrobi Fuentes	Caso de Uso probado Registrar nombre del rol. Se aplicaron dos pruebas	Una PC con la aplicación instalada que cumplía todos los requisitos de hardware y software	5/05/2007	2

Los errores encontrados de forma general fueron los siguientes:

- ✓ Error en la ejecución al eliminar alguna entrada.

- ✓ Hay mensajes que no emiten ningún texto aclaratorio.
- ✓ Cuando se trata de eliminar algún elemento que estuvo seleccionado

y que aún se muestren sus datos, se genera una excepción que, aunque es manipulada por el sistema en parte, no es controlada totalmente.

2.9 Resultados de las Pruebas realizadas al sistema SIGIA

La fase de prueba es importante para cualquier software, ya que con ella se verifica la calidad con la que se trabajó en el proceso de desarrollo del sistema. Los requisitos indispensables para lograr que las pruebas sean un éxito son:

- Llevar toda la documentación pertinente al software.
- Deben quedar reflejados claramente el análisis, el diseño y los resultados de la ejecución del programa utilizando el conjunto de valores definidos en los datos de pruebas.

Esto da la posibilidad de llevar tanto el control de los casos que se van a aplicar en cada tipo de prueba, como de los errores que se van obteniendo al ir aplicándolas, así como también permite que con una buena documentación los desarrolladores tengan la posibilidad de corregir los errores de cada una de las partes del sistema que les corresponda antes de que pasen a otro paso del desarrollo del software.

En las pruebas de caja blanca para lograr tener una lista previa de errores se debe de tener en consideración los criterios de cada programador acerca de los casos de uso realizados por ellos de cada módulo, ellos saben donde se pueden encontrar puntos complejos propensos a errores, donde puede haber algún error aún no corregido.

En el caso de las pruebas de Caja Negra, se especifican los resultados alcanzados por cada módulo, y se verifican la lista de requisitos con la interfaz del sistema. Para lograr visualizar todos los errores del sistema se tienen que considerar cuales podrían ser los errores más propensos a ocurrir ya sea en la parte del código, la interfaz, los requerimientos o las entradas de datos. Esto puede hacerse de varias formas, una de ellas es listando defectos conocidos en la práctica del desarrollo,

generando suposiciones a partir de la naturaleza del software, teniendo en cuenta complejidad de algunos métodos, etc. De cualquier forma, tener una idea previa de los defectos que puedan existir será aprovechable para todos los encargados de probar software.

Para mostrar los resultados de la prueba de caja negra se utilizó la siguiente estructura:

Una tabla con los siguientes datos:

CP-CN-Módulo Nomencladores	**Dificultades y Errores**	**Resultado Esperado**	**Clasificación de los errores**	**Observaciones**

✓ La descripción de los errores deben de ser claras y precisas.

Los errores se clasifican en:

Funcional: cuando los valores de las salidas de la ejecución discrepan de los valores esperados.

Catastrófico: el caso en el que el sistema admita entradas no válidas y que el funcionamiento del sistema sea satisfactorio.

Mejora: Solicitud de una nueva característica o funcionalidad

2.9.1 Resultados de la prueba de Caja Negra del módulo Nomencladores

Las pruebas de Caja Negra se ejecutaron utilizando una PC con sistema operativo Windows XP profesional, con 496 Mbyte de RAM y un microprocesador Pentium (R) 4 CPU 2.80 GHz. Se corrió la aplicación donde estaban todas las interfaces generadas para cada uno de los módulos que se probaron. El sistema exigía una conexión a una Base de datos que se encontraba en un servidor con las mismas características de hardware que las PC terminales. Las pruebas fueron realizadas por un probador del grupo de desarrollo.

Los resultados de los errores comprenden los resultados de los casos diseñados para las entradas, los errores del diseño de interfaz, flujo básico y los errores de las pruebas de requisitos.

CP-CN-Módulo Nomencladores	Dificultades y Errores	Resultado Esperado	Clasificación de los errores	Observaciones
CPR 1.1 Registrar un nuevo grupo al sistema.	El sistema debe mostrar un mensaje de error el campo "nombre" solo admite letras. No lo muestra , lo admite En el campo "Código" hay que dar doble clip para introducir el nuevo código	Insatisfactorio Insatisfactorio Fue detectado a la hora de probar los campos de la interfaz usuario	La clasificación del error es de: Tipo Mejora	✓ Faltan muchos mensajes Informativos, para guiar al usuario en el uso del sistema. ✓ Deben señalar en la interfaz los campos que nunca deben ser dejados de llenar para así asegurar y agilizar mas y mejor el trabajo. Se debería de quitar Listar proveedor porque ya está en la interfaz principal de Editar Especialidad Proveedor.
CPR 2.1 Crear unidad de Medida.	La aplicación guarda datos incorrectos :en el Campo Nombre, admite números el sistema debe de mostrar un mensaje de error porque el campo	Insatisfactorio	Tipo Mejora	

	nombre debe admitir letras El sistema debe de mostrar un mensaje de error porque el campo Métrica es obligatorio. No deja opción para dejarlo vacío La aplicación guarda los datos entrados sin llenar campos obligatorios: Campo Descripción, Campo Activo	Insatisfactorio Insatisfactorio.Fue detectado durante la prueba	Tipo catastrófico Tipo funcional	
CPR 5.2: Crear un nuevo cliente tercero	No tuvo errores	Las pruebas fueron Satisfactorias		
CPR 5.4: Listar clasificación de Sección	No tuvo errores	Las pruebas fueron Satisfactorias		
CPR 6.1 Registrar sección de inventario	No tuvo errores	Las pruebas fueron Satisfactorias		
CPR 6.2 Modificar Sección	No tuvo errores	Las pruebas fueron Satisfactorias		
CPR 7.1 Registrar la información de un proveedor	El sistema no muestra un mensaje de error ,te pone el código que se puso anteriormente, cuando debe de mostrar un mensaje de error. Debe mostrar un mensaje de error	Insatisfactorio	Tipo Catastrófico	

	donde plantee que el nombre es una cadena de letras y no de números Muestra un mensaje de "error" la cadena contiene caracteres ilegales que no lo debe de mostrar porque todos los caracteres están escritos correctamente.	Insatisfactorio	Tipo Mejora	

2.9.2 Resultados de la prueba de Caja Negra del módulo Administración

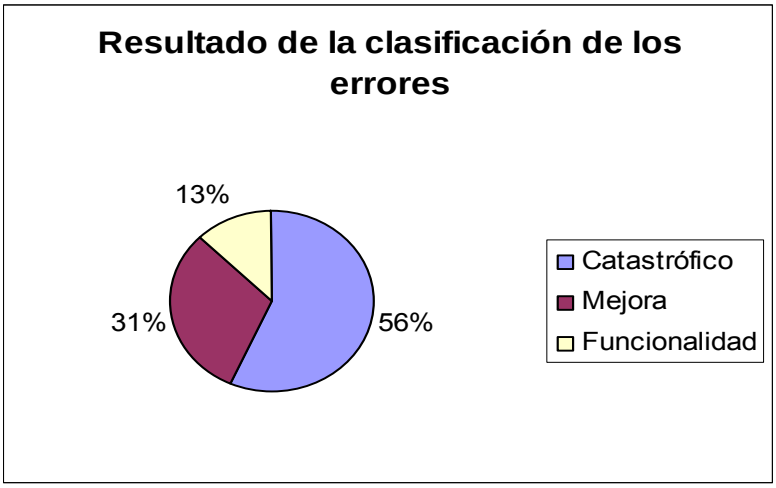
CP-CN-Módulo Nomencladores	**Dificultades y Errores**	**Resultado Esperado**	**Clasificación de los errores**	**Observaciones**
CPR 8.1 Crear un nuevo Terminal	El sistema guarda los datos en la base de datos pero es un error porque se deben de restringir la cantidad de caracteres. No se restringe en el campo nombre la cantidad de caracteres a entrar y en la base de datos es de 30 y lo restringe antes de llegar a esa cantidad.	Insatisfactorio Insatisfactorio	La clasificación del error es de: Tipo catastrófico	✓ Se deben de Señalar los campos significativos de la interfaz

2.9.3 Resultados de la prueba de Caja Blanca del módulo Nomencladores

Las pruebas de Caja Blanca se ejecutaron utilizando una PC con sistema operativo Windows XP profesional, con 496 MByte de RAM y un microprocesador Pentium (R) 4 CPU 2.80 GHz. Se corrió la aplicación donde estaban todas las interfases generadas para cada uno de los módulos que se probaron. El sistema exigía una conexión a una Base de datos que se encontraba en un servidor con las mismas características de hardware que las PC terminales. Las pruebas fueron realizadas por un probador del grupo de desarrollo.

CP-CB-Módulo Nomencladores	Dificultades y Errores	Clasificación de los errores	Observaciones
Caja Blanca al módulo Nomencladores	Los errores más frecuentes son: ✓Error en la ejecución al eliminar alguna entrada. ✓Hay mensajes que no emiten ningún texto aclaratorio. ✓Cuando se trata de eliminar algún elemento que estuvo seleccionado, y que aún se muestren sus datos, se genera una excepción que, es manipulada por el sistema.	La clasificación del error es de: Tipo catastrófico Tipo Mejora Tipo catastrófico	✓ Se deben de Señalar los campos significativos de la interfaz

Gráfica 1 Estado de los Resultados en por ciento



Este gráfico representa la cantidad de los errores en por ciento de las clasificaciones establecidas para los errores de los casos de pruebas. Solo se analizaron los resultados insatisfactorios de las pruebas realizadas.

Total de errores por módulos

Módulos Nomencladores

Se encontró un total de 51 errores entre no conformidades y resultados insatisfactorios de la aplicación.

Se encontró al aplicar las pruebas de Caja Negra al módulo un total de 48 errores entre resultados insatisfactorios de la aplicación y no conformidades.

En general en la aplicación de las pruebas de Caja Blanca a este módulo se encontró un total de 3 errores.

Módulo Administración

Se encontró un total de 11 error entre no conformidades y resultados insatisfactorios de la aplicación.

Se encontró al aplicar las pruebas de Caja Negra al módulo un total de 11 errores entre resultados insatisfactorios de la aplicación y no conformidades.

De los 13 casos de usos probados de caja negra y de estos 5 probados por el método de caja blanca permitieron obtener el siguiente resultado: 56% de los errores detectados en los casos de prueba están clasificados de catastróficos, 31% de mejora, y el 13% del tipo funcional.

De forma general, la aplicación de las pruebas a través del método de caja negra arrojó un total de 59 errores, mientras las pruebas de caja blanca arrojaron 3 errores.

2.9.4 Resultados del Sistema en general.

Dentro de los errores que se han detectado en el sistema, sólo una parte de ellos se pueden catalogar de catastróficos que es el caso en el que el sistema admite entradas no válidas y que el funcionamiento del sistema sea satisfactorio, aunque los más graves ocurren cuando se interrumpe la ejecución por esta causa y no se restauran los eventos necesarios para que la aplicación continúe, este último caso el sistema no presentó problemas.

Los errores fueron encontrados durante el funcionamiento de la interfaz del sistema en general, se puede decir que existen algunos aunque no todos de relevante importancia. La carencia de una ayuda del sistema que pueda guiar al usuario en su utilización es uno de ellos, además no se cuenta con mensajes lo suficientemente explicativos que puedan corregir y guiar al usuario a la hora de utilizar el sistema.

2.9.5 Comparación entre resultados reales y esperados.

Haciendo una breve comparación con los errores que se esperaban del sistema y los que se obtuvieron, podemos decir que el sistema realiza la mayor parte de las funciones que tiene implementadas solo contiene algunas fallas de validación y de tratamiento de errores, no se comenta el software como parte de una guía. Principalmente se esperaban algunos en el módulo nomencladores, específicamente en los eliminar de cada operación de las peticiones de cambio. También era probable que algunos eventos anidados emitieran fallas al restaurarse, y este no ocurrió, aunque en la mayoría de los casos no se respetaba la longitud de la cadena establecida en la base de datos, De forma genérica se puede plantear que se previeron la mayor parte de los resultados que generó el sistema.

2.10 Conclusiones parciales

- El desarrollo del Sistema de Gestión de Inventarios y Almacenes es de gran importancia porque integra varias funcionalidades independientemente del tipo de empresa que haga uso de el y estandariza el proceso de gestión de inventarios y almacenes en un software configurable a partir de las necesidades particulares de los clientes e incorpora todas las regulaciones que establece el Ministerio de Finanzas y Precios.
- El plan de pruebas descrito permitió obtener las estrategias, los recursos y el diseño de los casos de pruebas para los módulos que integran el producto SIGIA.
- Se describen las pruebas de Caja Negra y Caja, y se aplican las técnicas del camino básico y de partición de equivalencia a los casos de uso de los módulos de administración y de nomencladores de SIGIA.
- Se diseñaron y se desarrollaron casos de pruebas para los módulos de administración y nomencladores.
- Se realizó un análisis de los resultados obtenidos en las pruebas realizadas a cada uno de los casos de uso de los módulos nomencladores y administración del Sistema de Gestión de Inventario.

CONCLUSIONES GENERALES

- El aseguramiento de la calidad del software permite reducir notablemente los costos de producción e implantación, y proporciona mayor confianza en el cumplimiento de los requisitos del cliente, siendo las pruebas de software un elemento imprescindible para asegurar la calidad y para verificar el cumplimiento de los requisitos funcionales y los impuestos por el cliente.
- El diseño del plan de pruebas para el Sistema de Gestión de Inventario y Almacenes permitió describir las estrategias, recursos y planificación de las pruebas, y establecer un cronograma de pruebas para el SIGIA.
- Se estableció una metodología a aplicar en las pruebas de caja negra y en las de caja blanca para la ejecución de los casos de prueba.
- La aplicación de las pruebas a los casos de uso de los módulos Nomencladores y Administración permitió detectar los errores de los casos de uso y de la lógica del programa de la aplicación SIGIA.
- Al realizar los casos de prueba se obtuvo que el 56% de los errores detectados están clasificados de catastróficos, el 31% de mejora, y el 13% del tipo funcional.

RECOMENDACIONES

- Realizar un estudio más profundo acerca de la calidad de software, de su importancia en el proceso de desarrollo y acerca de la importancia de las pruebas de software como parte del aseguramiento de la calidad y las últimas tendencias internacionales en esta materia.
- Perfeccionar el plan de pruebas teniendo en cuenta la retroalimentación obtenida con la aplicación de las pruebas a los módulos Nomencladores y Administración.
- Usar las herramientas de pruebas estudiadas en el marco teórico, que permitirán reducir el tiempo de ejecución de las pruebas y aumentarán su eficiencia.

BIBLIOGRAFÍA

1. Alvear Rodríguez Tatiana, C. R. C. *Sistemas de Información para el Control de Gestión*. Departamento de Sistemas de Información y Auditoría. Chile, Universidad de Chile, 2005.
2. Cuatrecasas L. (2003). Gestión Competitiva de Stocks y Procesos de Producción Ed. Gestión 2000, pagina 33.
3. Cortés, O. H. G. Aplicación práctica del diseño de pruebas de software a nivel de programación, 20-Abril-2004.
4. C, L. Gestión Competitiva de Stocks y Procesos de Producción. en. 2000., E. G., 2003. pagina 33.p.
5. Cabet, M. *"A Software Complexity Measure",IEEE Trans , software Engineering.*, diciembre, 1976. p. 308-320 p.
6. Canales Mora Roberto: Desarrollo y arquitectura Java/J2EE. Asesoramiento Tecnológico Web. Calidad en el desarrollo de Software, CMMI. 2003-2005

   http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=cmmi
7. Fernández Carrasco Oscar M. Un enfoque actual sobre la calidad del software, septiembre-diciembre 1995.
8. García Joaquín. Calidad CMM-CMMI. Agosto 2005 http://www.ingenierosoftware.com/calidad/cmm-cmmi.php.
9. ISO 9000:2000 [2000]. Sistema de Gestión de la calidad. Principios Fundamentales y Vocabulario. Secretaria General ISO, Traducción certificada. Ginebra, Suiza.
10. ISO 9000:2005 [2005]. Sistemas de Gestión de la calidad. Principios Fundamentos y Vocabulario. Secretaría Central de ISO, Traducción certificada.

    Ginebra, Suiza.
11. IEEE, *IEEE Std. 610 Computer dictionary*, IEEE, 1990.
12. Jumela, F. G. Introducción a las ERPs. www.yoprogramo.com
13. Lovelle, J. M. C. *Calidad del Software* Calidad del Software Grupo GIDIS Universidad

    Nacional de la Pampa, 1999. p*. www.uniovi.es*.

14. [Myres, 2004] Glenford J. Myers, *The Art of Software Testing* 2a. Ed, Tom Badgett and Todd
    a. Thomas with Corey Sandler, John Wiley & Sons, Inc., Hoboken, New Jersey, 2004
15. Mañas, J. A. *Pruebas de Programas*, 28 de Febrero de 2002.
16. Pressman R. S. (1998). Ingeniería del software. Un enfoque práctico. 4ª Edición. McGrawHill.
17. Pressman, Roger S. (2002). Ingeniería del Software: Un enfoque práctico; Quinta edición. McGraw-Hill, Madrid.
18. Pressman Roger S. (2000): Software Engineering: "*A Practitioner's Approach*" European Adaptation). 5ta Edition. McGrawHill.
19. Paulk M.C., et al, Capability Maturity Model for Software, Version 1.1, Carnegie Mellon University, SEI-93-TR-024. 1993
20. Paulk M. C., Weber C.V., et. al., The Capability Maturity Model, Guidelines for Improving the Software Process, p. 180–191, CMU, SEI, 1999
21. Pulido Jose, Gestión de inventarios almacenes ,2006
22. Rational Unified Process. Rational Software Corporation. "Rational Unified Process". Versión 2003.06.00.65, Copyright 1987-2003.
23. Rodríguez, D. E. *Introducción a la Calidad del Software*, Septiembre, 2003.
24. Villalobos Hernández María de la luz, A. F. G. *Investigación sobre las prácticas de ingeniería de software en México*, Agosto, 2001.
25. Valladares, T. R. *JUnit*.
26. http://www.monografias.com/trabajos16/manual-de-inventario/manual-de-inventario.shtml#CONCEP, 9/noviembre/ del 2006
27. http://www.monografias.com/trabajos27/algoritmos-geneticos/algoritmos-geneticos.shtml 20/11/2006
28. http://www.yoprogramo.com/docs/Introduccion_a_las_ERP.pdf , revisado 13/diciembre/2006
29. http://www.sc.ehu.es/seweb/webcentro/cas/publica/numeros/n12/3.pdf, revisado, 9/Enero/2007.

30. http://www.giro.infor.uva.es/Publications/2004/MLC04/JENUI2004.pdf, revisado, 9/Enero/2007.

31. http://www.aiteco.com/caliaapp.htm, revisado 16 /enero / 2007

32. http://www.greensqa.com , revisado el 29/enero/2007

33. www.parquesoft.com , revisado 30/enero/2007.

34. http://www.adrformacion.com/cursos/calidad/calidad.html , revisado, 25/2/2007

35. http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html , revisado 14/3/2007

36. http://www.als-es.com/home.php?location=herramientas/entorno-pruebas, revisado, 14/3/2007

37. http://www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html, revisado el 20/3/2007.