Universidad de las Ciencias Informáticas
Facultad 3

Título: Pruebas para el software Administración Contable de los Registros y Notarías de la República Bolivariana de Venezuela

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Alianis La Hoz Guilarte

Tutor: Ing. Yanosky Rios La Hoz

Junio de 2007

DECLARACIÓN DE AUTORÍA

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

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

Alianis La Hoz Guilarte

_____

Yanosky Rios La Hoz

_____

AGRADECIMIENTOS

Agradezco a todas las personas (familiares, amigos y profesores) que de una forma u otra contribuyeron con mi formación, especialmente:

A mis padres porque siempre me ayudaron en todo lo que estuvo a su alcance, por haber soportado todas mis malacrianzas, por haberme apoyado siempre y lo más importante haber confiado en que el sueño de tantos años se está realizando.

A Jovette porque juntas hemos sido siempre una, porque siempre me ayudaste, me aconsejaste y apoyaste en todo. Porque siempre haz sido para mí, amiga, hermana y familia.

A mis amiguitas del pre porque también me ayudaron mucho y juntas compartimos momentos inolvidables (Daylin, Yohanka, Yeni y Milaine)

A Companioni, gracias por tus consejos y tu ayuda, aunque sea falsita, yo nunca te olvido.

A ti Valdi aunque nunca leas esto siempre voy a agradecer tu ayuda.

A ti Ame por ser amiga, hermana y familia. Por tus consejos. Porque cuando nos conocimos nos ayudamos mucho una a la otra.

A ti Danieluco tampoco te olvido.

A Pedro por haberme dado la oportunidad de crecerme.

A Ariadne y Evelyn, por su ayuda en estos años y en especial con la realización de este trabajo, gracias por todo.

A mi tutor yanosky por haberme guiado en esta etapa final.

A ti Monchi porque a pesar de llegar a mi vida ya casi en la recta final, me demostraste que sabías hacer de todo, me ayudaste en temas que no conocías, me ayudaste a ser organizada y responsable. Sin tu ayuda y apoyo todo me hubiera resultado mucho más difícil. No tengo palabras para agradecer todo lo que haz hecho por mí, por eso y más TE AMO.

DEDICATORIA

A mis padres por ser los ojitos que me han guiado siempre, porque ustedes son mi inspiración y sin su apoyo esto hoy no hubiera sido posible. Los amo mucho.

A mi hermana Elsy que forma parte de este lindo sueño.

A mi abuela Magdalena, por la confianza que me tuviste siempre.

A mis abuelos.

A Ruth tus consejos y apoyo me sirvieron de mucho.

A ti Erito que también te quiero mucho.

A Nury y Chiquito que también les debo mucho.

A ti Monchi por tu dedicación y entrega hacía mí, por todos tus desvelos y paciencia, TE AMO.

A mi familia en general.

Resumen

El desarrollo de un software implica una serie de actividades de producción donde la probabilidad de cometer al menos un error, es muy alta. Los problemas pueden comenzar partiendo de fallos que se hayan cometido en el levantamiento de requisitos de dicho sistema, lo que trae consigo que estos existan también en los pasos de diseño y desarrollo.

El presente trabajo realiza un análisis profundo acerca de los distintos tipos de pruebas, herramientas y metodologías, de las que se hizo una selección para emplearlas en el proceso de prueba desarrollado al módulo Administración Contable de los Registros y Notarías de la República Bolivariana de Venezuela, se analizaron herramientas que pueden ser empleadas para automatizar las pruebas en caso deseado, se realiza un estudio de la metodología empleada para la elaboración de algunos artefactos de prueba de manera manual y también se mencionan herramientas con las que estos se pueden hacer de forma automatizada. Para realizar un adecuado proceso de pruebas se elaboran artefactos como: plan de prueba, estrategia de prueba, casos de prueba, datos de prueba, configuración del entorno de prueba y evaluación de las pruebas.

Palabras Claves

Proceso de pruebas, plan de prueba, casos de prueba, estrategia de prueba.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Con la gran evolución de las Tecnologías de la Informática y las Comunicaciones, es necesario el desarrollo y aplicación de metodologías que influyen en la calidad de los productos de software, esto es una necesidad de primer orden para aquellas empresas, universidades, compañías y otras entidades que desarrollan software, cada vez más los errores que estos presentan repercuten en su prestigio. Es por ello la necesidad de desarrollar un adecuado proceso de pruebas, dentro del cual se distinguen hitos como: planificación, diseño, implementación, ejecución y evaluación de las pruebas. Este proceso comienza con la elaboración de un plan de pruebas, esta etapa es muy importante ya que define el marco de trabajo sobre el cual se van a desarrollar las pruebas y las actividades necesarias para la ejecución de las mismas, y concluye con la evaluación de las pruebas.

Situación Problémica

En la Facultad 3 de la Universidad de las Ciencias Informáticas se está desarrollando una solución informática para la informatización de los procesos desarrollados en las oficinas de los Registros y Notarías de la República Bolivariana de Venezuela. El proyecto tiene como cliente al Ministerio del Interior y Justicia de esta nación, el mismo está compuesto de diferentes módulos: Mercantil, Inmobiliario, Servicio Autónomo y Administración Contable. El desarrollo de este último módulo contará con las funcionalidades necesarias para garantizar que a través de él se realice toda la gestión contable y financiera de los Registros y Notarías de la República Bolivariana de Venezuela, facilitando mediante la información que brinda, el análisis de los factores económicos de la entidad. El sistema a desarrollar es muy complejo, para su construcción se cuenta con un grupo de estudiantes y profesores comprometidos con el desarrollo de este producto para que sea entregado en tiempo al cliente. Como en todo desarrollo, no están exentas las posibilidades de cometer errores en las distintas etapas de construcción del sistema, y dada la envergadura e importancia de dicho software se hace muy importante llevar a cabo pruebas que permitan detectar la mayor cantidad de defectos posibles, dado estas circunstancias se plantea el siguiente problema:

Problema a resolver

¿Cómo aplicar un adecuado proceso de pruebas en el módulo Administración Contable de los Registros y Notarías de la República Bolivariana de Venezuela?

Objeto

Proceso de pruebas de software.

Objetivo general

Llevar a cabo un adecuado proceso de pruebas en el módulo Administración Contable de los Registros y Notarías de la República Bolivariana de Venezuela, que permita encontrar la mayor cantidad de errores posibles.

Objetivos específicos

1. Elaborar un plan de prueba donde se definan las actividades y tareas necesarias para el proceso de pruebas.
2. Elaborar una estrategia de prueba que integre las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software.
3. Elaborar los datos de prueba y diseñar los casos de prueba para probar la funcionalidad del software.
4. Elaborar la configuración del entorno de prueba para conocer todos los recursos de hardware y software necesarios para la ejecución de las pruebas.
5. Realizar la evaluación de las pruebas para evaluar la cobertura de los casos de prueba y analizar los defectos encontrados.

Campo de acción

Pruebas de software en el módulo Administración Contable.

Hipótesis

Si se lleva a cabo un adecuado proceso de pruebas en el módulo Administración Contable de los Registros y Notarías de la República Bolivariana de Venezuela entonces se podrá detectar la mayor cantidad de errores posibles.

Tareas de la investigación

Para alcanzar los objetivos trazados y demostrar la hipótesis establecida se acometerán las siguientes tareas:

- Estudiar y describir el estado del arte de las pruebas de software según las tendencias y metodologías existentes.
- Estudiar la metodología RUP la cual será empleada para la elaboración de los artefactos seleccionados.
- Estudiar cómo elaborar los artefactos necesarios para realizar un adecuado proceso de pruebas.
- Realizar entrevistas a profesionales vinculados al tema de las pruebas de software.

El presente trabajo cuenta con tres capítulos:

En el capítulo 1 se realiza un estudio del estado del arte acerca de las pruebas de software, así como de las principales tendencias, metodologías y tecnologías existentes que se pueden emplear para la realización de pruebas de software en la actualidad.

En el capítulo 2 se elaboran los artefactos necesarios para el proceso de prueba de software al módulo Administración Contable como son: el plan de prueba, la estrategia de prueba, los casos de prueba, los datos de prueba y la configuración del entorno de prueba.

En el capítulo 3 se ejecutan las pruebas diseñadas y se realiza la evaluación de las pruebas efectuadas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se realiza un estudio acerca del estado del arte de las pruebas de software, y además de las principales tendencias y tecnologías que se pueden aplicar para la elaboración de las mismas. Citándose las propuestas seleccionadas para la realización de las pruebas al módulo Administración Contable de los Registros y Notarías de la República Bolivariana de Venezuela.

1.2. Tendencias

El desarrollo de un software implica una serie de actividades de producción donde la probabilidad de aparecer al menos una falla es muy común. Los errores pueden comenzar a aparecer desde el inicio del proceso de elaboración de los requisitos, así como en los pasos posteriores del diseño y desarrollo. La prueba de software es un elemento importantísimo para el aseguramiento de la calidad del sistema y a su vez representa una revisión final de las especificaciones, del diseño y de la codificación. Es una actividad práctica que se debe incorporar en cada equipo de desarrollo de software, que requiere de una enorme cantidad de tiempo y esfuerzo (1).

La creciente inclusión del software como un elemento más de muchos sistemas y la importancia de los costos asociados a un fallo del mismo, han motivado a la creación de pruebas más minuciosas y bien planificadas. La prueba es un proceso de ejecución de un programa con la intención de descubrir errores, donde no se puede asegurar la ausencia de defectos sino que se pueden demostrar la existencia de defectos en el software.

A continuación se realiza un análisis de las estructuras de pruebas: La prueba de unidades se plantea a pequeña escala, consiste en ir probando uno a uno los diferentes módulos que constituyen una aplicación (2).

Las pruebas de integración y de aceptación son de mayor escala, que pueden llegar a dimensiones industriales cuando el número de módulos es muy elevado, o la funcionalidad que se espera del programa es muy compleja.

Las pruebas de integración se centran en probar la coherencia semántica entre los diferentes módulos, tanto de semántica estática (se importan los módulos adecuados; se llama correctamente a los procedimientos proporcionados por cada módulo), como de semántica dinámica (un módulo recibe de otro lo que esperaba). Normalmente estas pruebas se van realizando por etapas, englobando progresivamente más y más módulos en cada prueba.

Las pruebas de integración se pueden empezar en cuanto se tienen pocos módulos, aunque no terminarán hasta que se disponga de la totalidad. En un diseño descendente (top-down) se empieza a probar por los módulos más generales; mientras que en un diseño ascendente se empieza a probar por los módulos de base.

El planteamiento descendente tiene la ventaja de estar siempre pensando en términos de la funcionalidad global; pero también tiene el inconveniente de que para cada prueba hay que "inventarse" algo sencillito (pero fiable) que simule el papel de los módulos inferiores, que aún no están disponibles.

El planteamiento ascendente evita tener que escribirse módulos ficticios, pues se van construyendo pirámides más y más altas con la información que se va teniendo. Su desventaja es que se centra más en el desarrollo que en las expectativas finales del cliente.

Estas clasificaciones no son las únicas posibles. Por ejemplo, en sistemas con mucha interacción con el usuario es frecuente codificar sólo las partes de cada módulo que hacen falta para una cierta funcionalidad. Una vez probada, se añade otra funcionalidad y así hasta el final. Esto da lugar a un planteamiento más "vertical" de las pruebas. A veces se conoce como "codificación incremental".

Por último, las pruebas de aceptación son las que se plantea el cliente final, que decide qué pruebas va a aplicarle al producto antes de darlo por bueno y pagarlo. De nuevo, el objetivo del que prueba es encontrar los fallos lo antes posible, en todo caso antes de pagarlo y antes de poner el programa en producción.

Técnicas de pruebas de software

Cualquier producto de ingeniería puede ser probado de una de estas formas:

1. Conociendo la funcionalidad específica para la cual fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa (3).
2. Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que “todas las piezas encajen”, o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

La primera técnica se denomina prueba de caja negra y la segunda se denomina prueba de caja blanca.

Prueba de caja negra:

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, que se produce una salida correcta y que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. Se obtienen conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del software.

Técnicas empleadas en las pruebas de caja negra

Particiones de equivalencia:

Muchas veces lo que aparentemente son varios errores pueden agruparse como si fueran el mismo. Si una variable es de tipo numérico y se le está asignando un caracter, no es necesario probar con todos los caracteres del código ASCII, se entiende que entre ellos hay una equivalencia. De ese modo, sólo se provee una serie finita de valores para cada condición de entrada (4):

- Un rango: un valor dentro del rango y dos fuera (uno por encima y otro por debajo).
- Un número de valores: un conjunto del número de valores especificados, un conjunto con menos valores y otro con más.
- Un valor: Todos los valores posibles y uno no posible.
- Un dato lógico: Un valor de condición Verdadero y otro Falso.

Análisis de los valores límites:

Es un refinamiento de la técnica anterior. Se basa en la idea de que siempre será más fácil encontrar un error en los límites del rango de valores que en el centro, por lo que se hacen más pruebas con valores límites (un valor por encima, el justo y un valor por debajo) y menos con los valores centrales.

Grafos causa-efecto:

Representan gráficamente las pruebas comentadas hasta ahora, dotando de formalidad al planteamiento.

Pruebas de comparación:

En los casos en que la aplicación es crítica a veces se generan dos versiones de la misma y se prueban de forma independiente, asegurando de ese modo que los resultados no sólo son correctos, sino que son idénticos en cualquier circunstancia.

Prueba de los valores típicos de error (conjetura de errores):

Se prueban los valores que se sospecha que pueden causar errores. Dichos errores son generalmente identificados en base a la experiencia del analista.

Prueba de datos imposibles:

Son los datos que figuran como inviables en la especificación de la aplicación, mereciendo, por tanto, una prueba dedicada.

Prueba de caja blanca:

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

Tipos de pruebas de caja blanca

De estructura de datos locales:

Se centra en el estudio de las variables del programa. Busca que toda variable esté declarada, que no haya variables con el mismo nombre declaradas local y globalmente, que haya

referencias a todas las variables y, para cada variable, analiza su comportamiento en comparaciones.

De Cobertura Lógica:

De Cobertura de Sentencias:

Comprueba que todas las sentencias se ejecuten al menos una vez.

De Cobertura de Decisión:

Ejecuta casos de prueba de modo que cada decisión se pruebe al menos una vez a Verdadera y otra a Falsa.

De Cobertura de Condición:

Ejecuta un caso de prueba a Verdadero y otro a Falso por cada condición, teniendo en cuenta que una decisión puede estar formada por varias condiciones.

De Cobertura de Condición/Decisión:

Se realizan las pruebas de cobertura de condición y las de decisión a la vez.

De Condición Múltiple:

Cada decisión multicondición se traduce a una condición simple, aplicando posteriormente la cobertura de decisión.

Prueba de Complejidad Ciclomática:

Propuesta por Tom McCabe y también llamada Número Ciclomático de McCabe. Para analizar este tipo de prueba antes se presentan algunos conceptos como:

Camino: Secuencia de todas las instrucciones de un programa de principio a fin.

Cobertura de caminos: el método de prueba ha de ser elegido después de trazar un camino para la prueba.

Prueba unitaria

La prueba unitaria es un procedimiento utilizado para validar que un módulo de código fuente funciona apropiadamente. Son comprobaciones que se le hacen a las unidades lógicas del software. Se verifica que una unidad funciona correctamente por sí misma, sin tener en cuenta las relaciones que pueda tener con otras partes del sistema. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las Pruebas de

Integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión (5).

Para que una prueba unitaria sea buena se deben cumplir los siguientes requisitos:

- Automatizable: no debería requerirse una intervención manual. Esto es especialmente útil para integración continúa.
- Completas: deben cubrir la mayor cantidad de código.
- Repetibles o Reutilizables: no se deben crear pruebas que sólo puedan ser ejecutadas una sola vez. También es útil para integración continua.
- Independientes: la ejecución de una prueba no debe afectar a la ejecución de otra.
- Profesionales: las pruebas deben ser consideradas igual que el código, con la misma profesionalidad que la documentación.

Aunque estos requisitos no tienen que ser cumplidos a cabalidad, se recomienda seguirlos o de lo contrario las pruebas pierden parte de su función.

El objetivo de las pruebas unitarias es aislar, cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas (6):

1. Fomentan el cambio: Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (esto es llamado refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
2. Simplifica la integración: Puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.
3. Documenta el código: Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.
4. Separación de la interfaz y la implementación.
5. Los errores están más acotados y son más fáciles de localizar: dado que se tienen pruebas unitarias que pueden descubrirlos.

Es importante darse cuenta de que las pruebas unitarias no descubrirán todos los errores del código. Por definición, sólo prueban las unidades por sí solas. Por lo tanto, no descubrirán errores de integración, problemas de rendimiento y otros problemas que afectan a todo el sistema en su conjunto. Además, puede no ser trivial anticipar todos los casos especiales de entradas que puede recibir en realidad la unidad de programa bajo estudio. Las pruebas unitarias son más efectivas si se usan en conjunto con otras pruebas de software.

Las pruebas unitarias hacen uso de las técnicas de caja negra y hacen uso intensivo de las técnicas de prueba de caja blanca.

Pruebas de integración

Las pruebas de integración se llevan a cabo durante la construcción del sistema, involucran a un número creciente de módulos y terminan probando el sistema como conjunto (7).

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

Estructurales de integración:

Son similares a las pruebas de caja blanca; pero trabajan a un nivel conceptual superior. En lugar de referirse a sentencias del lenguaje, se refiere a llamadas entre módulos. Se trata pues de identificar todos los posibles esquemas de llamadas y ejercitarlos para lograr una buena cobertura de segmentos o de ramas.

Funcionales de integración:

Son similares a las pruebas de caja negra. Aquí se trata de encontrar fallos en la respuesta de un módulo cuando su operación depende de los servicios prestados por otro(s) módulo(s). Según se va acercando al sistema total, estas pruebas se van basando más en la especificación de requisitos del usuario.

Las pruebas finales de integración cubren todo el sistema y pretenden cubrir plenamente la especificación de requisitos del usuario. El manual de usuario, también se utiliza para realizar pruebas hasta lograr una cobertura aceptable.

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

Pruebas de Sistema

Uno de los objetivos de las pruebas del sistema es verificar que el comportamiento externo del sistema software satisface los requisitos establecidos por los clientes y futuros usuarios del mismo. A medida que aumenta la complejidad de los sistemas software y aumenta la demanda de calidad, se hacen necesarios procesos y métodos que permitan obtener buenos conjuntos de pruebas del sistema (8).

Dentro de la prueba de sistema se pueden desarrollar distintos tipos de pruebas en función de los objetivos de las mismas, como por ejemplo: pruebas funcionales, pruebas de usabilidad, pruebas de rendimiento, pruebas de seguridad, entre otras.

En general, las pruebas de sistema consisten en identificar todos los caminos de ejecución posibles y generar una prueba por cada camino. Dichas pruebas se completan con datos de prueba, el resultado esperado, las precondiciones y poscondiciones de cada requisito funcional.

Pruebas de Aceptación

Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, sino una vez pasada todas las pruebas de integración por parte del desarrollador (9).

La experiencia muestra que aún después del más cuidadoso proceso de pruebas por parte del desarrollador, quedan una serie de errores que sólo aparecen cuando el cliente se pone a usarlo. Decir que los requisitos no estaban claros, o que el manual es ambiguo no deja satisfecho al cliente.

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

Las pruebas beta vienen después de las pruebas alfa, y se desarrollan en el entorno del cliente, un entorno que está fuera de control. Aquí el cliente se queda a solas con el producto y trata de encontrarle fallos (reales o imaginarios) de los que informa al desarrollador.

Las pruebas alfa y beta son habituales en productos que se van a vender a varios clientes. Algunos de los potenciales compradores se prestan a estas pruebas bien por ir entrenando a su personal con tiempo, bien a cambio de alguna ventaja económica (mejor precio sobre el producto final, derecho a mantenimiento gratuito, a nuevas versiones etc.). La experiencia muestra que estas prácticas son muy eficaces.

Otros tipos de pruebas

Recorridos (walkthroughs)

Quizás es una técnica más aplicada en control de calidad que en pruebas. Consiste en sentar alrededor de una mesa a los desarrolladores y a una serie de críticos, bajo las órdenes de un moderador que impida un recalentamiento de los ánimos. El método consiste en que los revisores se leen el programa línea a línea y piden explicaciones de todo lo que no está claro. Puede que simplemente falte un comentario explicativo, o que detecten un error auténtico o que simplemente el código sea tan complejo de entender/explicar que más vale que se rehaga de forma más simple. Para un sistema complejo pueden hacer falta muchas sesiones (10).

Aleatorias (random testing)

Ciertos autores consideran injustificada una aproximación sistemática a las pruebas. Alegan que la probabilidad de descubrir un error es prácticamente la misma si se hacen una serie de pruebas aleatoriamente elegidas, que si se hacen siguiendo las instrucciones dictadas por criterios de cobertura (caja negra o blanca).

Como esto es muy cierto, probablemente sea muy razonable comenzar la fase de pruebas con una serie de casos elegidos al azar. Esto pondrá de manifiesto los errores más patentes. No obstante, pueden permanecer ocultos otros errores que sólo se muestran ante entradas muy precisas.

Si el programa es poco crítico (una aplicación personal, un juego entre otras) puede que esto sea suficiente. Pero si se trata de una aplicación militar o con riesgo para vidas humanas, es de todo punto insuficiente.

Solidez (robustness testing)

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

Por ejemplo, en un sistema que admite una serie de órdenes (commands) se deben probar los siguientes extremos:

- Órdenes correctas, todas y cada una.
- Órdenes con defectos de sintaxis, tanto pequeñas desviaciones como errores de bulto.
- Órdenes correctas, pero en orden incorrecto, o fuera de lugar.
- La orden nula (línea vacía, una o más).
- Órdenes correctas, pero con datos de más.
- Provocar una interrupción (BREAK o lo que corresponda al sistema soporte) justo después de introducir una orden.
- Órdenes con delimitadores inapropiados (comas, puntos, etc.).
- Órdenes con delimitadores incongruentes consigo mismos.

Prestaciones (performance testing)

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

Conformidad u Homologación (conformance testing)

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

Las pruebas de caja negra, que se le hacen a un producto para detectar discrepancias respecto a una norma de las descritas en el párrafo anterior se denominan de conformidad u homologación.

Suelen realizarse en un centro especialmente acreditado al efecto y, si se hacen satisfactoriamente, el producto recibe un sello oficial que dice: "homologado".

Interoperabilidad (interoperability tesing)

En el mismo escenario del punto anterior, programas de comunicaciones que deben permitir que dos ordenadores se entiendan, aparte de las pruebas de conformidad se suelen correr una serie de pruebas, también de caja negra, que involucran dos o más productos, y buscan problemas de comunicación entre ellos.

Regresión (regression testing)

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

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

Las pruebas de regresión son particularmente espectaculares cuando se trata de probar la interacción con un agente externo. Existen empresas que viven de comercializar productos que "graban" la ejecución de una prueba con operadores humanos para luego repetirla cuantas veces haga falta "reproduciendo la grabación". Y, obviamente, deben monitorizar la respuesta del sistema en ambos casos, compararla, y avisar de cualquier discrepancia significativa.

1.3. Metodologías

1.3.1 ExtremeProgramming

Extreme Programming o programación Extrema, es una de las metodologías llamadas “ágiles”, para el desarrollo de proyectos de software. Se basa en los principios de la simplicidad, la comunicación, la retroalimentación y el coraje para implicar a todo el equipo (y a los usuarios o clientes) en la gestión del proyecto. En 1996, Kent Back y Ward Cunningham pusieron en práctica una nueva metodología primando la simplicidad y evitando los hábitos que convertían las cosas fáciles en

difíciles durante el desarrollo de un proyecto en DaimlerChrysler. El resultado fue la metodología Extreme Programming o XP (11).

Los procesos y prácticas de esta metodología están basados en la experiencia de equipos de desarrollo, y en los errores cometidos o encontrados una y otra vez al utilizar metodologías más tradicionales. Sin embargo, algunas de estas propuestas son bastante chocantes y otras incompatibles con grandes organizaciones o grandes proyectos. Extreme Programming, puede dividirse en cuatro principios sobre los que se va iterando hasta que el proyecto ha finalizado (el cliente aprueba el proyecto). Estas fases o principios son planificación, diseño, desarrollo y pruebas. En el caso de tener una fecha de entrega muy próxima, la construcción de las pruebas unitarias va a ahorrar mucho más tiempo del que se invierte programándolas, buscando pequeños fallos y la protección contra ellos de forma permanente durante el resto del desarrollo. A veces existe la tendencia a pensar que las pruebas unitarias pueden hacerse durante los últimos tres meses de desarrollo. Esto es erróneo, ya que sin esas pruebas, el desarrollo se alarga esos tres meses, y quizá más.

Los test unitarios ayudarán también a la refactorización, ya que asegurarán que los cambios que se hayan introducido en la iteración actual no afecten a la funcionalidad. Cuando se encuentre un fallo en las pruebas de aceptación con el cliente, o durante el uso, se debe crear una prueba de unidad que lo compruebe. Así se asegura que no vuelva a surgir en siguientes iteraciones. Las pruebas de aceptación se crearán a partir de las historias de usuario. Una historia puede tener una o varias pruebas, según sea la funcionalidad que hay que probar. El cliente es responsable de definir las pruebas de aceptación, que deberían ser automatizables en la medida de lo posible. Estas pruebas son del tipo “caja negra”, en el sentido de que únicamente definen el resultado que debe tener el sistema ante unas entradas concretas. Las pruebas de aceptación que no tengan éxito generarán nuevas tareas para la próxima iteración, afectando así a la velocidad del proyecto, y proporcionando además una puntuación del éxito o fracaso de cada historia de usuario, o de cada equipo de trabajo.

1.3.2 Desarrollo Basado en Pruebas (TDD)

Desarrollo guiado por pruebas, o Test-driven development (TDD) es una técnica de programación enfatizada en la programación extrema. Esencialmente la técnica implica el escribir primero sus pruebas y luego implementar el código para ejecutarla. La meta del desarrollo conducido por las pruebas es lograr una rápida retroalimentación e implementa el "ilustrar la línea principal" al hacer un programa. Muchos enfatizan que el desarrollo conducido por las pruebas es sobre todo un método de diseño de software, no solo un método de pruebas (12).

Para que funcione el desarrollo guiado por pruebas, el sistema tiene que ser lo suficientemente flexible como para permitir el testeo automático de software, usando casos de prueba que devuelven un simple verdadero o falso en su evaluación. Estas propiedades permiten una rápida retroalimentación en el diseño y la corrección. Frameworks como JUnit y NUnit proveen de un mecanismo para manejar y ejecutar conjuntos de pruebas automatizadas.

1.3.3 Proceso Unificado de Desarrollo de Software

El Proceso Unificado de Rational o RUP (Rational Unified Process), es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado (13).

Sus principales características son:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.

- Verificación de la calidad del software.

RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, está centrado en la arquitectura y es guiado por los casos de uso. Incluye artefactos (son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al concluir cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

Inicio: se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos.

Elaboración: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos, y además queda definida la arquitectura base.

Construcción: se concentra en la elaboración de un producto totalmente operativo y eficiente, y el manual de usuario.

Transición: se implanta el producto al cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

RUP propone que en cada una de las fases las pruebas se comporten de la siguiente forma:

- Inicio: El desarrollo del prototipo exploratorio de demostración; no requiere la elaboración de pruebas.
- Elaboración: Probar los componentes ejecutables que se han implementado y que deben corresponderse con la arquitectura básica de la aplicación.
- Construcción: Desarrollar los casos de prueba y procedimientos de prueba para hacerlos.
- Transición: El producto en su entorno de operación por lo que es probado por usuarios reales.

Flujo de trabajo

A menudo existen ciertos malentendidos que se pueden deducir equivocadamente:

- El que desarrolla el software no debe entrar en el proceso de prueba.
- El software debe ser “puesto a salvo” de extraños que puedan probarlo de forma despiadada.
- Los encargados de la prueba solo aparecen en el proyecto cuando comienzan las etapas de prueba.

Todo esto es incorrecto, el que desarrolle el software es responsable de probar los módulos o subsistemas asegurándose que cada una lleva a cabo la funcionalidad para el que fue diseñado. Una vez que la arquitectura del software esté completa, entra en juego un grupo independiente de prueba que elimina el conflicto de interés que para el desarrollador él tiene que probar su propio producto (14).

Aunque las pruebas pueden ser realizadas por usuarios finales y otros trabajadores del equipo, RUP define que los roles que intervienen en el flujo de prueba son los siguientes:

1. Diseñador de pruebas: Planifica, diseña y evalúa los resultados de la prueba.
2. Ingeniero de componentes: Responsable de la elaboración de los componentes de prueba para los procedimientos que pueden ser automatizados.
3. Ingeniero de pruebas de integración: Realizan las pruebas de integración.
4. Ingeniero de pruebas de sistema: Realiza las pruebas de sistema.

En la fase de transición en las pruebas participa un equipo mixto de desarrolladores y clientes reales.

RUP propone tres grandes pasos para realizar las pruebas:

1. Planificar las pruebas de integración; tanto de integración para cada construcción, como del sistema, dentro de cada iteración.
2. Diseñar e implementar las pruebas con la elaboración de casos de prueba; así como evaluar la utilización de algún software que permita automatizar las pruebas.

3. Realizar las pruebas según lo planificado anteriormente y arreglar los defectos detectados como resultado de la ejecución de las pruebas.

El principal objetivo de la prueba es realizarlas y evaluarlas como se describe en el modelo de pruebas. Los responsables de las pruebas (Ingenieros de pruebas en RUP) inician esta tarea planificando el esfuerzo de prueba en cada iteración. La secuencia de acciones que RUP propone como método consiste en:

1. Planificar prueba:

Planifica los esfuerzos de prueba en una iteración describiendo una estrategia de prueba, estimando los requisitos para el esfuerzo de la prueba (recursos humanos, sistemas necesarios, etc.) y planificando el esfuerzo de la prueba.

2. Diseñar prueba:

Se trata de identificar y describir los casos de prueba para cada construcción, e identificar y estructurar los procedimientos de prueba especificando cómo realizar los casos de prueba.

3. Implementar la prueba:

El propósito de implementar la prueba es automatizar en la medida de lo posible los procedimientos de prueba creando componentes de prueba, algo que no siempre es posible o incluso rentable, dado que no todos los procedimientos de prueba pueden ser automatizados.

4. Realizar pruebas de integración:

Se realizan las pruebas de integración necesarias para cada una de las construcciones creadas en una iteración y se recopilan los resultados de las pruebas.

5. Realizar pruebas al sistema:

Se realizan las pruebas del sistema, que pueden comenzar cuando las pruebas de integración indiquen que el sistema satisface los objetivos de calidad de integración fijado en el plan de prueba de la iteración actual.

6. Evaluar prueba:

Se evalúan los resultados de la prueba comparando los resultados obtenidos con los objetivos esbozados en el plan de prueba.

Pruebas de software empleando esta metodología

Durante la actividad de prueba se verifica el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas al cliente. Los objetivos de las pruebas son, planificar las pruebas necesarias en cada iteración, diseñar e implantar las pruebas creando los casos de prueba que especifican qué probar y realizar dichas pruebas manejando los resultados sistemáticamente. La actividad de pruebas en RUP se extiende prácticamente a lo largo de toda la vida del proyecto, dado que a medida que se va alcanzando una comprensión alta del contexto del sistema durante el análisis, se pueden ir diseñando los casos de prueba que verificarán el correcto funcionamiento del sistema una vez esté construido. Así durante la implementación, se realizan pruebas unitarias de los componentes que se van construyendo nada más terminados.

- **Modelo de Prueba**

Describe principalmente como se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema. Puede también describir como se prueban ciertos aspectos específicos del sistema, como por ejemplo, la usabilidad de su interfaz, o la efectividad del manual de usuario. Se compone de una colección de:

- **Casos de prueba**

Un caso de prueba especifica una forma de probar el sistema, incluyendo la entrada o el resultado con la que se ha de probar y las condiciones bajo las que ha de probarse. Los casos de prueba comunes son los que especifican como probar un caso de uso o un escenario concreto de un caso de uso.

- **Procedimiento de prueba**

  Especifica como realizar uno o varios casos de prueba o partes de estos. Un mismo procedimiento puede ser aplicado a varios casos de prueba.

- **Componentes de prueba**

  Un componente de prueba automatiza uno o varios procedimientos de prueba o partes de ellos.

Casos de Prueba

En la Ingeniería del software, los casos de prueba o Test Case son un conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio (15).

Se pueden realizar muchos casos de prueba para determinar que un requisito es completamente satisfactorio. Con el propósito de comprobar que todos los requisitos de una aplicación son revisados, debe haber al menos un caso de prueba para cada requisito a menos que un requisito tenga requisitos secundarios. En ese caso, cada requisito secundario deberá tener por lo menos un caso de prueba. Algunas metodologías como RUP recomiendan el crear por lo menos dos casos de prueba para cada requisito. Uno de ellos debe realizar la prueba positiva de los requisitos y el otro debe realizar la prueba negativa.

Si la aplicación es creada sin requisitos formales, entonces los casos de prueba se escriben basados en la operación normal de programas de una clase similar.

Lo que caracteriza un escrito formal de caso de prueba es que hay una entrada conocida y una salida esperada, los cuales son formulados antes de que se ejecute la prueba. La entrada conocida debe probar una precondición y la salida esperada debe probar una poscondición.
Bajo circunstancias especiales, podría haber la necesidad de ejecutar la prueba, producir resultados, y luego un equipo de expertos evaluaría si los resultados se pueden considerar como "Correctos". Esto sucede a menudo en la determinación del número del rendimiento de productos nuevos. La primera prueba se toma como línea base para los subsecuentes ciclos de pruebas/lanzamiento del producto.
Los casos de prueba escritos, incluyen una descripción de la funcionalidad que se probará, la cuál es tomada ya sea de los requisitos o de los casos de uso, y la preparación requerida para asegurarse de que la prueba pueda ser dirigida.
Los casos de prueba deben satisfacer los siguientes criterios:

- Reducir, en un coeficiente mayor que uno, el número de casos de prueba adicionales.
- Que digan algo sobre la presencia o ausencia de clases de errores.

Plan de pruebas

El plan de prueba tiene como propósito contornear y comunicar el esfuerzo de prueba para un tiempo determinado. Su objetivo principal es obtener la aceptación y aprobación de los involucrados en el proceso y esfuerzo de prueba. Se debe evitar el detalle que no sería entendido, o sería considerado inaplicable por los involucrados en el proceso y esfuerzo de prueba. Como segundo punto, el plan de prueba forma el marco dentro del cual el equipo que realizará la prueba trabajará para el tiempo determinado (16).
El plan de prueba captura los siguientes elementos informativos: La definición de las metas y de los objetivos del esfuerzo de prueba dentro del alcance de la iteración (o del proyecto), una explicación de la estrategia que será utilizada, los recursos y tiempo requeridos.

Estrategia de pruebas

La Estrategia de Prueba de software describe el acercamiento y los objetivos generales de las actividades del proceso de prueba. Incluye los niveles de prueba (unidad, integración, y sistema)

y las clases de prueba (función, funcionamiento, carga, tensión) que se realizarán. Define los entregables que se producirán y elementos como (17):

- Técnicas y herramientas de prueba que serán utilizadas.
- Qué criterios de culminación y éxito de la prueba serán utilizados. Por ejemplo, los criterios pudieron permitir que el software progrese la prueba de unidad, cuando un determinado porciento de los casos de prueba se hayan ejecutado con éxito. Otro criterio es el de cobertura de casos de pruebas según los niveles de defectos encontrados.
- Planificación de las pruebas.
- Diseño de casos de prueba.
- Implementación de las pruebas.
- Ejecución de las pruebas.
- Resultados de las pruebas.

Se definen roles como:

1. Administrador de pruebas incluye las siguientes responsabilidades:

- Planeamiento y logística.
- Acepta la misión.
- Identifica los motivadores.
- Adquiere los recursos apropiados.
- Evalúa la eficacia del esfuerzo de la prueba.

2. Analista de pruebas identifica y define las pruebas específicas que se conducirán. Incluye las siguientes responsabilidades:

- Identificar las ideas de prueba.
- Define los detalles de prueba.
- Elabora el documento de peticiones de cambio.
- Determina los resultados de prueba.
- Evalúa la calidad del producto.

3. Diseñador de pruebas define la técnica de acercamiento a la implementación del esfuerzo de prueba. Incluye las siguientes responsabilidades:

- Definir el acercamiento de prueba.
- Define la arquitectura de automatización de prueba.
- Verifica técnicas de prueba.
- Define elementos del testability.
- Estructura la implementación de prueba.

4. Probador implementa y ejecuta las pruebas. Incluye las siguientes responsabilidades:

- Implementa las pruebas y las suites pruebas.
- Ejecuta las suites pruebas.
- Analizar y recuperar las fallas de prueba.

5. Administrador de pruebas de sistema asegura el entorno de pruebas. Incluye las siguientes responsabilidades:

- Administrar las pruebas de sistema.
- Instala y soporta el acceso, y la recuperación, de configuraciones del entorno de prueba y laboratorios de prueba.

6. Administrador de base de datos asegura el entorno de datos de prueba (base de datos). Incluye las siguientes responsabilidades:

- Soporta la administración de datos de prueba y de base de datos.

7. Diseñador identifica y define las operaciones, los atributos, y las asociaciones de las clases de prueba. Incluye la siguiente responsabilidad:

- Define las clases de prueba requeridas para apoyar requisitos de testability según lo definido por el equipo de prueba.

8. Implementador realiza las pruebas de unidad, clases de prueba y paquetes de prueba. La responsabilidad incluye:

- Crea los componentes de prueba requeridos para soportar los requisitos de testability según lo definido por el diseñador.

Tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software.

1.4. Herramientas

- JUnit: Entorno de pruebas para Java creado por Erich Gamma y Kent Beck.
- TestNG: Creado para suplir algunas deficiencias en JUnit.
- JTiger: Basado en anotaciones, como TestNG.
- SimpleTest: Entorno de pruebas para aplicaciones realizadas en PHP.
- Rational TestManager.
- NUnit: Migración del entorno JUnit para lenguajes de la plataforma .NET.

1.4.1 Junit

JUnit es un conjunto de librerías creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java. Permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa 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 (18).

JUnit es también 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.

1.4.2 TestNG

TestNG es un marco de prueba inspirado de JUnit y de NUnit, pero se pueden introducir algunas nuevas funcionalidades que lo hagan más de gran alcance y más fácil utilizar, por ejemplo:

Anotaciones del JDK 5 (el JDK 1.4 también se apoya con las anotaciones de JavaDoc). Configuración flexible de la prueba (19).

- Ayuda para la prueba data-driven.
- Ayuda para los parámetros.
- Permite la distribución de pruebas en las máquinas auxiliares.
- Es un modelo de gran alcance de ejecución.
- Apoyado por una variedad de herramientas y de plug-ins (eclipse, IDEA, Maven, etc.).
- Encaja BeanShell para la flexibilidad adicional.
- Omite las funciones del JDK para el tiempo de pasada y registrar (ningunas dependencias).
- Métodos dependientes para la prueba del servidor de uso.
  TestNG se diseña para cubrir categorías de pruebas como: unidad, funcional, integración, entre otras.

1.4.3 JTiger

El framework JTiger para el estándar Java 2 edición 1.5, es una solución del open source que proporciona una abstracción robusta y una buena característica para desarrollar y ejecutar casos de prueba de unidad. Los casos de prueba metadatos y la documentación de prueba de unidad se expresa con anotaciones (20).

1.4.4 SimpleTest

En el contexto del desarrollo ágil, el código de prueba está junto al código fuente y ambos se escriben simultáneamente. En este contexto SimpleTest apunta ser una solución completa de prueba del desarrollador PHP y se llama “simple” porque debe ser fácil de utilizar y de extender. Incluye todas las funciones típicas de JUnit y de los puertos de PHPUnit, pero también agrega objetos falsos. También tiene cierta funcionalidad de JWebUnit (21).

Esta herramienta es utilizada para crear plan de pruebas y casos de pruebas.

1.4.5 Rational TestManager

A continuación se proporciona información sobre la herramienta Rational TestManager para la creación de un plan de prueba (22):

- Define el acercamiento de la prueba
- Identifica los motivadores de prueba.
- Identifica los blancos de prueba.

Rational TestManager es la consola central para la administración, ejecución y divulgación de la prueba. Construido para la extensibilidad, apoya los acercamientos de la prueba manual a los varios paradigmas automatizados incluyendo la regresión de prueba de unidad y funcional.

1.4.6 Nunit

NUnit es una de esas herramientas utilizada para escribir y ejecutar pruebas en .NET, es un framework desarrollado en C# que ofrece las funcionalidades necesarias para implementar pruebas en un proyecto. Además provee una interfaz gráfica para ejecutar y administrar las mismas. También se utiliza para realizar pruebas unitarias a .NET (23).

Lo que hace tan efectiva la TDD (Test Driven Development) es la automatización de las pruebas de programador, (programmer unit tests).

El hecho de utilizar TDD implica 3 acciones: escribir las pruebas, escribir el código que debe pasar las pruebas y refactorizar para eliminar el código duplicado. La primera se lleva a cabo de

una manera sencilla y simple utilizando NUnit, esta soporta los lenguajes bases de .NET como C#, J#, VB y C++.

NUnit es una herramienta que se encarga de analizar ensamblados generados por .NET, interpretar las pruebas inmersas en ellos y ejecutarlas. Utiliza atributos personalizados para interpretar las pruebas y provee además métodos para implementarlas. En general, NUnit compara valores esperados y valores generados, si estos son diferentes la prueba no pasa, caso contrario la prueba es exitosa.

NUnit carga en su entorno un ensamblado y cada vez que lo ejecuta, o mejor, ejecuta las pruebas que contiene, lo recarga. Esto es útil porque se pueden tener ciclos de codificación y ejecución de pruebas simultáneamente, así cada vez que se compile no tiene que volver a cargar el ensamblado al entorno de NUnit si no que este siempre obtiene la última versión del mismo. NUnit ofrece una interfaz simple que informa si una prueba falló, pasó o fue ignorada.

NUnit basa su funcionamiento en dos aspectos, el primero es la utilización de atributos personalizados. Estos atributos le indican al framework de NUnit que debe hacer con determinado método o clase, es decir, estos atributos le indican a NUnit como interpretar y ejecutar las pruebas implementadas en el método o clase.

El segundo son las aserciones, que no son más que métodos del framework de NUnit utilizados para comprobar y comparar valores.

1.5. Automatización de pruebas

La automatización de pruebas se ha convertido en una actividad fundamental para garantizar despliegues rápidos sin perder en calidad (24).

- Acelerar y maximizar el retorno de su inversión en automatización.
- Encontrar errores, agujeros de seguridad y problemas de rendimiento antes de que lleguen al cliente.
- Construir unas bases sólidas para automatizar el trabajo de pruebas en las versiones siguientes y futuros sistemas.
- Le ayudarán en el cumplimiento de fechas de entrega ajustadas, asegurando la satisfacción del usuario y reduciendo los riesgos.

La automatización reduce al mínimo el esfuerzo en pruebas funcionales y de regresión, permite pruebas consistentes y repetibles, facilita la ejecución de pruebas complejas y en definitiva permite ganar en tiempo de mercado.

1.5.1 Automatización de Pruebas - QA Wizard

Automatiza pruebas funcionales. Brinda el soporte a pruebas en ambientes Web, Windows y en aplicaciones de Java. Realiza la validación de datos, permitiendo verificar los resultados durante y después de una serie de pruebas. Los defectos o errores descubiertos durante las pruebas pueden ser incorporados automáticamente a TestTrack para su manejo y resolución. Los scripts pueden ser controlados en Surround o Visual SourceSafe para facilitar las pruebas (25).

Se requiere de QA Wizard Cuando los responsables de probar las aplicaciones utilizan demasiado tiempo en probar las rutinas existentes en lugar de aprovecharlo en probar las nuevas funcionalidades recientemente incorporadas. Cuando se requiere de una mayor cantidad de pruebas para garantizar la calidad. Cuando el recurso humano cuesta más que el mismo software. Cuando los encargados de hacer las pruebas pueden pasar por alto problemas o sólo encuentran errores superficiales.

Algunos de sus beneficios son el aumento en la productividad, reducción de las pruebas manuales y de errores humanos, mayor cobertura de prueba en menor tiempo y reducción de tiempo en la liberación de la aplicación.

1.6 Conclusiones

En este capítulo se hizo un análisis del estudio del arte de las pruebas, acerca de las principales tendencias y tecnologías que se pueden aplicar para la elaboración de las pruebas. Se definió emplear la metodología RUP y realizar las pruebas de sistema, empleando específicamente la técnica de caja negra para comprobar que el software esté funcionando correctamente. Para ello es necesario diseñar casos de prueba con el objetivo de verificar que:

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

- La integridad de la información externa se mantiene.

Además realizar un plan de prueba que capture los elementos informativos como: una explicación de la estrategia que será utilizada, los recursos y tiempo requeridos. Se va a elaborar una estrategia de prueba de software que describa el acercamiento y los objetivos generales de las actividades del proceso de prueba, definir los entregables que se producirán y elementos como:

- Técnicas y herramientas de prueba que serán utilizadas.
- Criterios de culminación y éxito de la prueba.
- Diseño de casos de prueba.

Se crearán los datos prueba y la configuración del entorno de prueba. Todos estos artefactos serán elaborados mediante la Ayuda de la Herramienta Case Rational.

CAPÍTULO 2: ARTEFACTOS DEL PROCESO DE PRUEBA

2.1. Introducción

En el presente capítulo se realiza la elaboración de los artefactos según la solución propuesta para lograr desarrollar un adecuado proceso de prueba al módulo Administración Contable de los Registros y Notarías de la República Bolivariana de Venezuela.

2.2. Plan de pruebas

Propósito

Este plan de prueba para el módulo Administración Contable contiene los siguientes objetivos:

1. Identificar la información existente del proyecto y los elementos de software que deben ser probados.
2. Identificar los casos de uso más importantes para diseñar los casos de prueba, y los requisitos de alto nivel que serán ejecutados.
3. Describir la estrategia de prueba que se empleará.

Visión del negocio

El módulo Administración Contable tiene como objetivo llevar a cabo todo el control de las operaciones realizadas en cada uno de los registros tanto Inmobiliarios como Mercantiles.
Este sistema cuenta con las funcionalidades necesarias para garantizar que a través de él se realice toda la gestión contable financiera de un registro, facilitando mediante la información que brinda, el análisis de los factores económicos de la entidad. Tiene diferentes procesos de la gestión contable financiera agrupando funcionalidades más específicas.

Alcance

Se asume emplear la técnica de la caja negra para el desarrollo de las pruebas, donde los casos de prueba pretenden demostrar que las funciones de cada caso de uso del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta.

Esta prueba no tiene prácticamente en cuenta la estructura interna del software, y además se apoya en la especificación de requisitos del módulo. El problema con las pruebas de caja negra no suele estar en el número de funcionalidades proporcionadas por el módulo (que siempre es un número muy limitado en diseños razonables); sino en los datos que se le pasan a estas funciones. El conjunto de datos posibles suele ser muy amplio (por ejemplo, un entero).

Para realizar estas pruebas existe una técnica algebraica llamada "clases de equivalencia", 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. Para la generación de estas clases no se puede armar un modelo, pero se pueden seguir las siguientes pautas como guía utilizable para la creación de cada clase. Por ejemplo:

- si un parámetro de entrada está comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, dentro y por encima del rango.
- si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, dentro y por encima del rango.
- si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
- si una entrada es booleana, hay 2 clases: verdadero o falso.

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Identificación del proyecto

La siguiente tabla identifica la documentación y disponibilidad necesaria para desarrollar el plan de prueba:

Documento (Versión / Fecha)	**Creado o Disponible**	**Recibido**	**Autor**
Documento de Arquitectura	si	si	Luis Enrique Carrazana

Especificaciones de Casos de Uso	si	si	Yaumarys Pino Cueto
Glosario de Términos	si	si	Yaumarys Pino Cueto
Requisitos Funcionales	si	si	Yaumarys Pino Cueto
Modelo del Negocio	si	si	Yaumarys Pino Cueto
Manual de Usuario	si	si	Yaumarys Pino Cueto

Pruebas de requerimientos

Los requerimientos de software deben tener una explicación clara, precisa y completa del problema que facilite el análisis de errores y la generación de casos de prueba. Un asunto de gran importancia es asegurar la corrección, coherencia y exactitud de los requisitos.

A continuación se muestran los artículos (casos de uso, requisitos funcionales y requisitos no funcionales) que se han identificado para probar. Esta lista representa qué será probado.

Pruebas de funcionalidad

Verificar la funcionalidad de las operaciones correspondientes al caso de uso Gestionar Registro de Comprobantes (seleccionar cuentas contables según el documento primario que se esté procesando, guardar comprobante, un comprobante que no esté balanceado no se puede guardar, crear un comprobante contable manual, ver todos los comprobantes de un período determinado, imprimir satisfactoriamente todos los documentos de salida del sistema, un comprobante que no sea de Administración Contable pueda abrirse o anularse, no se pueda abrir un comprobante contable que haya sido anulado).

- Verificar la funcionalidad de las operaciones correspondientes al caso de uso Cerrar Cuentas Nominales (seleccionar cuentas nominales, visualizar los pases asociados al comprobante, cerrar cuentas nominales y que se genere el comprobante contable).

- Verificar la funcionalidad de selección según el rango de fechas para obtener como resultado el nivel de cuenta contable correspondiente a las características seleccionadas y además que se identifiquen las cuentas con saldo inverso a la naturaleza, correspondiente al caso de uso Visualizar Reporte Contable Balance de Comprobación.
- Verificar la funcionalidad de seleccionar según el rango de fechas el nivel de las cuentas de ingreso y de egreso, y además que se identifiquen las cuentas con saldo inverso a la naturaleza, correspondiente al caso de uso Visualizar Reporte Contable del Estado de Ganancias o Pérdidas.
- Verificar la funcionalidad de selección según el rango de fechas para obtener como resultado el nivel de cuenta contable correspondiente a las características seleccionadas y además que se visualice el documento de salida, correspondiente al caso de uso Visualizar Reporte Contable del Balance General.
- Verificar la funcionalidad de crear el fondo de caja para pagos menores, con la cuenta contable asociada y el saldo que dispone, y además que se visualice el mismo, correspondiente al caso de uso Definir el Fondo de Caja.
- Verificar la funcionalidad de las operaciones correspondientes al caso de uso Registrar Egresos de Cajas (adicionar nuevos documentos de egresos de pagos menores, con su importe y conceptos de contrapartida asociados, confirmar o guardar nuevos documentos con las operaciones de anular o modificar cuando el documento se encuentra confirmado y eliminar o confirmar cuando el documento se encuentra en estado guardado, visualizar los documentos que se encuentran en estado de edición confirmación o anulación en la fecha seleccionada).
- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Gestionar Factura (búsqueda de un proveedor, agregar una nueva factura, confirmar o guardar una nueva factura, liquidar(únicamente cuando se le haya realizado un pago parcial), anular, modificar y eliminar facturas cuando se cumplan las condiciones siguientes: si la factura es nueva y nunca ha sido confirmada esta puede ser eliminada, confirmada o guardada; si la factura ya ha sido confirmada esta puede ser liquidada,

anulada o modificada; si la factura ya ha sido liquidada o anulada a esta no se le podrá realizar ninguna operación. Que la entrada del número de factura y el número de control pueda ser número y letra, y la entrada del importe Sin derecho a Crédito Fiscal sea mayor o igual que cero y menor que el importe de la factura).

- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Gestionar Notas de Débito y Crédito (crear, guardar, confirmar y eliminar una nota si esta se encuentra en estado de edición, modificar y anular una nota si esta se encuentra en estado de confirmación, buscar las notas de un proveedor, buscar las notas creadas en una fecha determinada, abrir una nota, y asociar una factura a una nota).
- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Configurar las Cuentas Bancarias (adicionar cuentas bancarias, si previamente se definieron las cuentas contables correspondientes, visualizar la conciliación de cada una de las cuentas bancarias y advertir si existen diferencias entre los saldos según el banco y según el libro, eliminar una cuenta bancaria si la misma no posee operaciones asociadas a ella, y visualizar el reporte de conciliación bancaria).
- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Registrar Asignación de Presupuesto (insertar un ingreso presupuestario, guardar o confirmar el documento, no permitirá adicionar un documento con el mismo número y que este pertenezca al mismo tipo de documento y a la misma cuenta bancaria, editar un nuevo documento y con posterioridad la posibilidad de guardarlo, confirmarlo o eliminarlo, una vez que esté confirmado, debe permitir anularlo o modificarlo, y una vez que esté anulado, no debe permitir realizar ninguna acción sobre el mismo).
- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Registrar Depósitos por Clasificar (insertar un documento de depósitos por clasificar, un documento en edición pueda ser guardado, confirmado o eliminado, una vez que esté confirmado debe permitir que se modifique o se anule y debe permitir que se clasifique, es decir que se le asocie un comprobante bancario).

- Verificar funcionalidad de alerta en caso de asociar un comprobante contable de número diferente al del documento de depósito por clasificar.
- Verificar que se cumpla la funcionalidad donde los importes de ambos documentos (Depósito por clasificar y comprobante bancario) sean iguales cuando se clasifique un depósito.
- Verificar la funcionalidad donde una vez que el documento esté clasificado o anulado no se realice alguna acción sobre el mismo.
- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Registrar Créditos Adicionales (insertar dos documentos con el mismo número y que estos pertenezcan al mismo tipo de documento y a la misma cuenta bancaria, editar un nuevo documento y con posterioridad la posibilidad de guardarlo o confirmarlo, si el documento se encuentra en estado guardado debe permitir confirmarlo o eliminarlo, si el documento está confirmado debe permitir anularlo o modificarlo, si el documento está anulado no debe permitir realizar ninguna acción sobre el mismo, debe generar un comprobante contable cuando se confirme, se anule o se modifique el documento, y debe permitir adicionar, eliminar y modificar las cuentas de contrapartida).
- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Registrar Otros Ingresos (insertar un documento de otros ingresos, provenientes de conceptos diferentes a los comprobantes bancarios y asignaciones presupuestarias, guardar o confirmar el documento y con posterioridad si fue confirmado, se podrá anular o modificar, no permitirá adicionar un documento con el mismo número y que este pertenezca al mismo tipo de documento y a la misma cuenta bancaria, editar un nuevo documento y con posterioridad la posibilidad de guardarlo, confirmarlo o eliminarlo, una vez que el documento esté cancelado no debe permitir realizar ninguna acción sobre el mismo, y además adicionar, eliminar y modificar las cuentas de contrapartida).
- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Registrar Pago de Obligaciones (seleccionar el proveedor o el ente recaudador al cual se le asociará el documento de Pago de Obligaciones o los pagos anticipados, según

corresponda, captar el número, tipo de documento, importe y cuenta bancaria que ampara los pagos de obligaciones que se ejecutan, visualizar las obligaciones pendientes de pago del proveedor o del ente recaudador seleccionado para facilitar su procesamiento, pagar obligaciones, eliminarlas o modificarlas según sea el caso, incluso hacer un pago parcial de una obligación, visualizar las obligaciones pagadas por el documento, editar un nuevo documento de pago de obligaciones y con posterioridad la posibilidad de guardarlo, confirmarlo o eliminarlo si este está en edición, una vez que el documento esté confirmado debe permitir anularlo o modificarlo, y una vez que el documento esté anulado no debe permitir realizar ninguna acción sobre el mismo).

- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Registrar Pago al Contado (buscar los proveedores, crear nuevos documentos, confirmar o guardar nuevos documentos, si el documento está en estado guardado se puede eliminar o confirmar el mismo, y en caso de que sea confirmado solamente se podrá anular o modificar, visualizar documentos que se encuentren en estado de edición, confirmación o anulación en una fecha seleccionada, abrir un documento ya creado, y visualizar, adicionar, eliminar y modificar los conceptos de contrapartidas).
- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Registrar Reembolso de Caja (editar un nuevo documento y con posterioridad la posibilidad de guardarlo, confirmarlo o eliminarlo, una vez que el documento este confirmado se puede anular o modificar, si está anulado no se puede realizar ninguna acción sobre el mismo, y se genera un comprobante contable cuando se confirme, se anule o se modifique el documento).
- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Registrar Transferencia a Oficinas (editar un nuevo documento y con posterioridad la posibilidad de guardarlo, confirmarlo o eliminarlo, una vez que esté confirmado se puede anular o modificar, si está anulado no se puede realizar ninguna acción sobre el mismo, no se puede repetir el mismo número de documento para una misma cuenta bancaria y un mismo tipo de documento, se genera un comprobante contable cuando se confirme, se

anule o se modifique el documento, adicionar, eliminar y modificar las cuentas de contrapartida, y actualizar el saldo de la cuenta bancaria una vez aceptada las operaciones que actualicen la base de datos).

- Verificar la funcionalidad de todas las operaciones correspondientes al caso de uso Registrar Otros Egresos (editar un nuevo documento y con posterioridad la posibilidad de guardarlo, confirmarlo o eliminarlo, una vez que esté confirmado se puede anular o modificar, si está anulado no puede realizar alguna acción sobre el mismo, se genera un comprobante contable cuando se confirme, se anule o se modifique el documento, y adicionar, eliminar y modificar las cuentas de contrapartida).
- Verificar la funcionalidad de selección de un estado de cuenta y la modificación del mismo, siempre y cuando este sea el último, correspondiente al caso de uso Crear Estados de Cuentas.
- Verificar la funcionalidad de las operaciones correspondientes al caso de uso Gestionar Retenciones (buscar un proveedor, buscar un documento de retención por el tipo de retención, por la fecha, por la forma de pago seleccionada y por el estado de documento, visualizar todos los documentos de retenciones creados a través de un reporte, crear un nuevo documento de retenciones, seleccionar la forma de pago deseada para gestionar la retención que pueden ser: Egreso de Caja, Pago al Contado, Pago de Obligaciones y la opción Otras Formas de Pago. Confirmar, eliminar o guardar un documento que se encuentre en estado de Edición, modificar o anular un documento que se encuentre en estado de Confirmación, asociar el documento de pago a la retención que se está gestionando, confirmar, eliminar o guardar un documento que se encuentre en estado de Edición, modificar o anular un documento que se encuentre en estado de Confirmación, asociar el documento de pago a la retención que se esta gestionando, y asociar una factura o no al documento de gestión de retención si la forma de pago seleccionada es Otras Formas de Pago).
- Verificar la funcionalidad de las operaciones relacionadas con al edición de comprobantes de retención (adicionar una nueva operación al comprobante, modificar y/o eliminar una

operación del comprobante, y crear un comprobante de retención), correspondiente al caso de uso Editar Comprobante de Retención.

Estrategia de Prueba

En esta sección se presenta el enfoque a utilizar para probar el sistema software. En la sección anterior se ha descrito qué elementos del sistema software serán probados, y en esta se define cómo se realizarán las pruebas.

Objetivo de la Prueba	Asegurar la navegación correcta de la aplicación, la entrada de datos, su procesamiento y recuperación.
Técnica	Ejecutar cada caso de uso, incluyendo los flujos alternos de cada caso de uso y utilizando datos válidos e inválidos con el objetivo de verificar que: • Se muestren los resultados previstos cuando se utilizan datos válidos. • Se exhibe el error apropiado y los mensajes de alerta cuando se utilizan datos inválidos. • Que los casos de uso estén funcionando como se esperaba.
Criterio de culminación	Todas las pruebas planificadas se han ejecutado. Todos los defectos identificados se han considerado.

Consideraciones	Ninguna.

Actividades del proceso de pruebas

Las actividades del proceso de pruebas para este sistema software son:

Hitos	Esfuerzo	Fecha Inicio	Fecha Fin
Planificación de las pruebas	1	Noviembre/2005	Noviembre/2005
Diseño de las pruebas	1	Diciembre/2006	Enero/2006
Implementación de las pruebas	1	Enero/2006	Febrero/2006
Ejecución de las pruebas	1	Febrero/2006	Junio/2006
Evaluación de las pruebas	1	Junio/2006	Junio/2006

Resultados de las pruebas

Del proceso de prueba se obtienen los siguientes documentos de desarrollo de software:

Documento de desarrollo de software	**Desarrollador**	**Revisión**	**Fecha**
Plan de prueba	Alianis La Hoz		Noviembre/2006
Estrategia de prueba	Alianis La Hoz		Noviembre/2006
Datos de prueba	Alianis La Hoz		Enero/2006
Casos de prueba	Alianis La Hoz		Enero/2006
Configuración del entorno de prueba.	Alianis La Hoz		Febreo/2006

Evaluación de pruebas	Alianis La Hoz		Junio/2006

Tareas de la etapa de pruebas

Las tareas que se realizan en cada una de las actividades son:

- **Planificación de las pruebas:**
    - Identificar los casos de uso más importantes para las pruebas.
    - Desarrollar la estrategia de pruebas.
    - Identificar los recursos necesarios para realizar las pruebas.
    - Generar el Plan de pruebas.
- **Diseño de las pruebas:**
    - Identificar y describir los casos de prueba.
- **Implementación de las pruebas:**
    - Establecer la configuración del entorno de prueba.
    - Desarrollar los datos de prueba.
- **Ejecución de las pruebas:**
    - Ejecutar los casos de prueba.
    - Verificar los resultados.
    - Registrar los defectos.
- **Evaluación de las pruebas:**
    - Evaluar la cobertura de los casos de prueba.
    - Analizar los defectos.
    - Crear el informe de evaluación de las pruebas.

2.3. Estrategia de pruebas

Propósito

El propósito de la estrategia de prueba para el ciclo de vida del módulo Administración Contable es que se defina el acercamiento general que será empleado para probar el software y para evaluar los resultados de lo que será probado.

Esta estrategia de prueba también apoya los siguientes objetivos específicos:

- Emplear la técnica de la caja negra.
- Serán probados los casos de uso más importantes del módulo, estos se mencionaron en el plan de pruebas.
- Definir los elementos entregables del proceso de prueba.

Entregables del proceso de prueba

A continuación se indican los entregables del proceso de prueba.

- Casos de Prueba.
- Datos de prueba o diccionario de datos.

Alcance

Se realizarán pruebas de caja negra con el objetivo de verificar:

- Que se muestren los resultados previstos cuando se utilizan datos válidos.
- Que se exhibe el error apropiado y los mensajes de alerta cuando se utilizan datos inválidos.
- Que los casos de uso estén funcionando como se esperaba.
- Que se cumpla con los requisitos del cliente.
- Que el diseño de pruebas permita encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo.

- Que se definan ciclos de prueba al final de los cuales se evalúen los resultados y se corrijan los errores encontrados con la actualización de la documentación del sistema.

Contexto y fondo del proyecto

El módulo Administración Contable tiene el objetivo de llevar a cabo todo un control de las operaciones realizadas en cada uno de los registros tanto de Inmobiliarios como Mercantiles.
Este sistema cuenta con las funcionalidades necesarias para garantizar que a través del mismo se realice toda la gestión contable financiera de un registro, facilitando mediante la información que brinda, el análisis de los factores económicos de la entidad.

Técnica.

Las pruebas de funcionalidad se centran en los requisitos de los casos de uso más importantes. 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, o sea, verificar la aplicación interactuando a través de las interfaces de usuario y analizando los resultados.

Objetivo de la técnica	Asegurar la navegación correcta de la aplicación, y verificar la aceptación, procesamiento y recuperación de datos.
Técnica	Ejecutar cada escenario del caso de uso o las funciones y características individuales, empleando datos válidos e inválidos, con el objetivo de verificar: • Que se muestren los resultados previstos cuando se utilizan datos válidos. • Que se exhibe el error apropiado y los mensajes de alerta cuando se utilizan datos inválidos.

	• Que los casos de uso estén funcionando como se esperaba. • Que se cumpla con los requisitos del cliente. • Que el diseño de pruebas permita encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo. • Que se definan ciclos de prueba al final de los cuales se evalúen los resultados y se corrijan los errores encontrados con la actualización de la documentación del sistema.
Herramientas Requeridas	Ninguna
Criterios del éxito	Son los criterios basados en las interfaces y las especificaciones del módulo desarrollado.
Consideraciones especiales	

Sistema

Los recursos del sistema para el proceso de prueba se encuentran en el epígrafe 2.6 “Configuración del Entorno de Prueba”.

Recursos humanos

Esta sección presenta los recursos humanos recomendados.

Recursos Humanos		
Rol	**Recursos mínimos recomendados**	**Responsabilidades o comentarios específicos**
Analista de pruebas	2	Identifica y define las pruebas específicas que se conducirán. Incluye las siguientes responsabilidades Define los detalles de prueba. • Determina los resultados de prueba. • Evalúa las pruebas.
Diseñador de pruebas	2	Define la técnica de prueba. Incluye las siguientes responsabilidades: • Elaborar el Plan de pruebas. • Generar la estrategia de pruebas. • Diseñar los Casos de prueba.
Probador	2	Ejecuta las pruebas Incluye las siguientes responsabilidades: • Ejecuta las pruebas. • Analizar y recuperar las fallas de prueba. • Documentar los defectos.

2.4. Casos de prueba

Introducción

Los siguientes casos de prueba especifican una forma de probar el sistema, incluyendo la entrada y salida con la que se ha de probar y las condiciones bajo las que ha de probarse. Se describen las clases válidas e inválidas que se pueden utilizar para realizar las pruebas del software.

Se realizaron los casos de pruebas según los veintidós casos de uso más importantes, y de ellos se tuvieron en cuenta sus requisitos.

A continuación se muestran algunos de los casos de prueba diseñados para la ejecución de las mismas.

Caso de Prueba 1

Verificar la creación de un comprobante contable manual.

Caso de uso 1

Gestionar Registro de Comprobantes

Propósito

Comprobar que el sistema permita seleccionar una cuenta contable según el documento primario que se esté procesando, y además que para poder guardar un comprobante contable debe tener al menos dos pases, el cual debe estar balanceado, o sea el valor del débito tiene que ser igual al valor del crédito, o la sumatoria de estos sea cero.

Descripción General del caso de uso

En esta funcionalidad se crea un nuevo comprobante, donde se insertan y seleccionan los datos requeridos.

Flujo Central

- Se muestra la interfaz principal de la aplicación, el usuario solicita al sistema crear un nuevo comprobante contable, para ello selecciona en el menú la opción: “Contabilidad”, “Registro de Comprobantes”, y luego presiona el botón “Nuevo”.
- El sistema muestra la interfaz “Editar Comprobante Contable” con los siguientes datos a llenar:
    - Descripción.
    - Cuenta o Descripción de la cuenta.
    - Débito.
    - Crédito.

- El sistema verifica que se hayan introducido todos los datos y además que las cuentas de débito y crédito estén balanceadas.
- El usuario acepta crear el comprobante.

Condiciones de ejecución

- Deben estar definidas las cuentas contables.
- Deben estar definidas las reglas contables.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado con datos válidos**	**Resultado Esperado con datos inválidos**	**Observación**
El usuario inserta los datos a llenar de forma correcta en los campos: Descripción (cadenas de caracteres, números	El usuario inserta los datos a llenar en los campos: Descripción (símbolos que no están	El sistema activa el botón Aceptar.	El sistema no permite introducir datos de este tipo.	Hasta que no estén introducidos todos los campos el sistema no activará el botón

y los símbolos que están definidos). Ejemplo: Comprobante de apertura Cuenta (escribir el número de cuenta) o Descripción de la cuenta (seleccionar). Ejemplo: 11102 o Inversiones temporales Débito (números mayor o igual que uno). Ejemplo:1500 Crédito (números mayor o igual que uno). Ejemplo:1500	definidos). Ejemplo: a!”· Cuenta (escribir cualquier número que no corresponda con alguna de las cuentas que están definidas, o caracteres y símbolos extraños) o Descripción de la cuenta (seleccionar). Ejemplo:9999 Débito (escribir caracteres, símbolos extraños o números menores o igual que cero). Ejemplo: -1500			Aceptar.

	Crédito (escribir caracteres, símbolos extraños o números menores o igual que cero). Ejemplo: -1500			
El usuario presiona el botón Aceptar.		El sistema crea el comprobante, mostrándose en la interfaz Registro de Comprobantes Contables.		

Caso de Prueba 2

Verificar el cierre de las cuentas nominales.

Caso de uso 2

Cerrar Cuentas Nominales

Propósito

Comprobar que se pueda realizar el cierre de las cuentas nominales y que se visualicen los pases asociados al comprobante.

Descripción General del caso de uso

Este caso de uso consiste en generar de forma automática un comprobante para llevar el saldo de las cuentas nominales a valor cero.

Flujo Central

Nota: Para poder visualizar la interfaz Cerrar Cuentas Nominales antes se tienen que configurar las reglas contables además del nomenclador Cuentas de oficina.

- El usuario solicita al sistema iniciar el cierre de las Cuentas Nominales.
- El sistema solicita editar los datos necesarios para cerrar las cuentas nominales.
  - Código Cuenta o Descripción de la cuenta contable.
  - Descripción del Comprobante.
- El usuario introduce los datos que el sistema solicitó.
- El usuario ordena generar el comprobante.
- El sistema muestra todos los pases asociados al comprobante generado.
- El usuario ordena guardar el comprobante.
- El sistema guarda el comprobante.
- Si el comprobante no es generado el sistema muestra un mensaje indicando que las cuentas nominales ya están cerradas.

Condiciones de ejecución

- Deben estar definidas las cuentas contables.
- Deben estar definidas las reglas contables.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado con datos válidos**	**Resultado Esperado con datos inválidos**	**Observación**
El usuario selecciona la Descripción del Comprobante (seleccionar), o	El usuario introduce el código de la Cuenta (escribir cualquier número que	El sistema muestra los pases asociados a esa cuenta y activa el	El sistema no permite introducir datos de	

introduce el Código de la Cuenta (escribir el número de cuenta). Ejemplo: 11102 o Inversiones temporales	no corresponda con alguna de las cuentas que están definidas, o caracteres y símbolos extraños) y la descripción del comprobante (símbolos extraños). 88808	botón Generar.	este tipo.	
El usuario presiona el botón Generar.		El sistema muestra un mensaje indicando si realmente se desea Cerrar las Cuentas Nominales.		
El usuario presiona el botón Aceptar.		El sistema cierra las Cuentas Nominales.		

Caso de Prueba 3
Verificar la selección de fechas para un nivel de cuenta contable determinado.

Caso de uso 3
Visualizar Reporte Contable Balance de Comprobación

Propósito

Comprobar la selección de fechas según el rango deseado con el objetivo de obtener como resultado el nivel de cuenta contable correspondiente a las características seleccionadas y además que se identifiquen las cuentas con saldo inverso a la naturaleza.

Descripción General del caso de uso

Este caso de uso consiste en realizar un Reporte Contable del registro mostrando en cualquier nivel de las cuentas los saldos de las mismas en el período contable analizado y acumulado en el Ejercicio Fiscal. Partiendo de la información que van generando los comprobantes, en este caso de uso se contempla la posibilidad de visualizar e imprimir el reporte.

Flujo Central

- El usuario solicita al sistema iniciar la visualización del Reporte Contable Balance de Comprobación.
- El sistema muestra los comprobantes contables.
- Si el usuario desea filtrar los comprobantes por varios parámetros el sistema solicita la selección de los siguientes datos:
- Nivel de la cuenta.
- Fecha desde.
- Fecha hasta.
- El usuario selecciona los datos.
- El sistema visualiza los comprobantes de acuerdo a la selección realizada.
- Si el usuario desea puede visualizar los comprobantes.

Condiciones de ejecución

- El sistema debe tener configurado los niveles de cuentas.
- Deben estar configuradas las reglas contables.

Clases Válidas	Clases Inválidas	Resultado Esperado con datos válidos	Resultado Esperado con datos	Observación

			inválidos	
El usuario selecciona la fecha del comprobante y el nivel de cuenta. Ejemplo: Rubro y 01/12/2006 a 31/12/2006		El sistema muestra los comprobantes contables correspondientes al nivel de cuenta y fecha seleccionado.		
El usuario presiona el botón Visualizar.		El sistema muestra el reporte de balance de comprobación.		
	El usuario selecciona la Fecha Hasta menor que la Fecha Desde. 31/12/2006 a 01/12/2006		El sistema no permite seleccionar la Fecha Hasta menor que la Fecha Desde.	
	El usuario inserta el nivel de cuenta.		El sistema solo permite seleccionar.	

Caso de Prueba 6

Verificar que se cree el fondo de caja.

Caso de uso 6

Definir el Fondo de Caja.

Propósito

Comprobar que se cree el fondo de caja para pagos menores, con la cuenta contable asociada y el saldo que dispone, y además que se visualice el mismo.

Descripción General del caso de uso

Este caso de uso consiste en definir el fondo de caja, la cuenta contable asociada al mismo y el saldo correspondiente en el momento en que se implanta el sistema.

Flujo Central

Nota: El fondo de caja se configura una sola vez en el sistema.

- El usuario solicita al sistema iniciar la configuración del fondo de caja.
- El sistema muestra los datos necesarios para la configuración del fondo de caja.
- Código del fondo.
- Nombre del fondo.
- Código cuenta caja o Descripción cuenta caja.
- El usuario edita o selecciona los datos.
- El usuario confirma la configuración del fondo de caja.
- El sistema guarda la configuración.

Condiciones de ejecución

- Deben existir cuentas contables.

- Deben estar definidas las reglas contables.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado con datos válidos**	**Resultado Esperado con datos inválidos**	**Observación**
El usuario introduce el Código del fondo (escribir un número mayor que cero), Ejemplo: 14116 el nombre del fondo (escribir cadena de caracteres y números) Ejemplo: Fondo de caja Código cuenta caja o Descripción cuenta caja. Ejemplo: 11102 o Inversiones temporales	El usuario introduce el Código del fondo (escribir un número menor o igual que cero), Ejemplo: -14116 El nombre del fondo (escribir símbolos) Ejemplo: fo¡”·do Código cuenta caja o Descripción cuenta caja. Ejemplo: 87889	El sistema guarda la configuración del fondo caja.	El sistema no permite introducir datos de este tipo.	El fondo de caja se realiza una vez, solo cuando instala la aplicación.

Caso de Prueba 7

Verificar los egresos de caja.

Caso de uso 7

Registrar Egresos de Caja.

Propósito

Verificar la funcionalidad de adicionar nuevos documentos de egresos de pagos menores, con su importe y conceptos de contrapartida asociados. Que se pueda confirmar o guardar nuevos documentos. Cuando el documento es guardado, el sistema debe permitir eliminar o confirmar el mismo, y en caso de que sea confirmado solamente se podrá anular o modificar. Y además comprobar que se visualicen los documentos que se encuentren en estado de edición, confirmación o anulación en una fecha seleccionada.

Descripción General del caso de uso

Este caso de uso consiste en realizar todas las operaciones de egresos que se ejecutan en la caja, las mismas están relacionadas únicamente con pagos menores.

Flujo Central

- El usuario solicita al sistema registrar los Egresos de Caja.
- El sistema muestra los datos del fondo de caja.
- El usuario ordena crear un nuevo documento.
- Si el usuario desea visualizar los documentos de un día determinado selecciona la fecha deseada.
- El sistema visualiza en la interfaz los documentos creados ese día.
- Si el usuario desea seleccionar un documento en estado de edición ordena confirmar el documento, inicia el caso de uso Confirmar Egreso de Caja.
- Si el usuario desea eliminar el documento, inicia el caso de uso Eliminar Documento.

- Si el usuario desea seleccionar un documento en estado de confirmación, ordena modificar el documento, inicia el caso de uso Modificar Egreso de caja.
- El usuario ordena modificar el documento, inicia el caso de uso Modificar Egreso de caja.
- Si el usuario desea anular el documento, inicia el caso de uso Anular Egreso de caja.
- El sistema solicita los datos necesarios para crear un nuevo documento:
- Importe.
- Descripción del movimiento.
- Acción a realizar.
- Importe a retener.
- El usuario edita o selecciona los datos.
- El usuario puede realizar varias operaciones.
- Si el usuario desea gestionar los conceptos de contrapartida, inicia el caso de uso Gestionar Conceptos de Contrapartida.
- El usuario puede realizar dos operaciones.
- Si el usuario desea guardar el documento, inicia el caso de uso Guardar documento.
- Si el usuario desea confirmar el documento, inicia el caso de uso Confirmar Egreso de Caja.

Condiciones de ejecución

- Debe estar definido el fondo de caja.
- Deben existir conceptos de contrapartida.
- El importe del documento debe ser menor o igual al fondo de caja.
- Deben estar definidas las reglas contables.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado con datos válidos**	**Resultado Esperado con datos inválidos**	**Observación**
El usuario introduce el importe (número	El usuario introduce el importe (número	• El sistema crea el	El sistema no permite	

mayor que cero), Ejemplo: 16000 la Descripción del movimiento (número o cadena de caracteres) Ejemplo: Egreso por papel y de manera opcional el importe a retener (número mayor que cero). Ejemplo: 1000 El usuario adiciona Conceptos de contrapartida introduciendo el importe (número mayor que cero). Ejemplo: 1400	menor o igual que cero), Ejemplo: -16000 la Descripción del movimiento (símbolos extraños) Ejemplo: re¡”· Y, de manera opcional el importe a retener (número menor o igual que cero). Ejemplo: -1000 El usuario adiciona Conceptos de contrapartida introduciendo el importe (número menor o igual que cero). Ejemplo: -1400	documento.	introducir datos de este tipo.	

Caso de Prueba 8

Verificar la funcionalidad de gestionar factura.

Caso de uso 8

Gestionar Factura

Propósito

Verificar la funcionalidad de todas las operaciones correspondientes a la gestión de factura (búsqueda de un proveedor, agregar una nueva factura, confirmar o guardar una nueva factura, liquidar(únicamente cuando se le haya realizado un pago parcial), anular, modificar y eliminar facturas cuando se cumplan las condiciones siguientes: si la factura es nueva y nunca ha sido confirmada esta puede ser eliminada, confirmada o guardada; si la factura ya ha sido confirmada esta puede ser liquidada, anulada o modificada; si la factura ya ha sido liquidada o anulada a esta no se le podrá realizar ninguna operación. Que la entrada del número de factura y el número de control pueda ser número y letra, y la entrada del importe Sin derecho a Crédito Fiscal sea >= 0 y menor que el importe de la factura).

Descripción General del caso de uso

Este caso de uso consiste en fijar las obligaciones de pago por factura recibida de los proveedores.

Flujo Central

- El usuario solicita al sistema Gestionar una factura.
- El sistema muestra las facturas creadas.
- El usuario ordena buscar un proveedor, edita el RIF del proveedor, ordena buscar el proveedor y el sistema visualiza los documentos asociados al proveedor.
- Si el usuario desea buscar la factura por fecha, debe seleccionar la fecha deseada.
- El sistema visualiza en la interfaz los documentos creados ese día.

- El sistema visualiza las facturas asociadas al proveedor.
- El usuario puede realizar varias operaciones.
- El usuario puede terminar el caso de uso.
- Si el usuario desea crear una nueva factura
- El sistema muestra los datos necesarios para crear una factura.
    - No. de Factura.
    - No. de Control.
    - Descripción del movimiento.
    - Fecha de emisión.
    - Importe.
    - Sin derecho a crédito Fiscal.
    - Base Imponible.
    - % Alícuota.
    - Fecha de vencimiento.
    - Código Cuenta o Descripción de la Cuenta.
    - Acción a realizar.
- El usuario edita o selecciona los datos.
- El usuario puede realizar varias operaciones.
- Si el usuario desea puede gestionar los conceptos de contrapartida
- El usuario ordena guardar el documento.
- Si el usuario desea confirmar el documento.
- Si el usuario desea abrir una factura en estado de edición.
- El usuario ordena abrir la factura.
- El usuario ordena confirmar la factura.
- Si el usuario desea puede eliminar una factura.
- Si el usuario desea abrir una factura en estado de confirmación.
- El usuario ordena abrir la factura.
- El usuario ordena modificar la factura.

- Si el usuario desea anular una factura.
- Si el usuario desea liquidar una factura.

Condiciones de ejecución

- Deben estar definidas las reglas contables.
- Deben estar definidas las cuentas contables de oficina.
- Deben estar definidos los conceptos de contrapartida.
- Debe estar definido el Nomenclador de Proveedores.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado con datos válidos**	**Resultado Esperado con datos inválidos**	**Observación**
El usuario inserta el RIF (número hasta ocho cifras). Ejemplo: V-12562354-3	El usuario inserta el RIF (número con más de ocho cifras). Ejemplo: V-125623522554-3	El sistema busca el proveedor.	El sistema no permite introducir datos de este tipo.	
El usuario crea una factura, introduciendo los siguientes campos: Nro de Factura (número mayor que cero y/o cadena de caracteres	El usuario crea una factura, introduciendo los siguientes campos: Nro de Factura (número menor o igual que cero y/o símbolos extraños),	El sistema crea la factura.	El sistema no permite introducir datos de este tipo.	

permitida), Ejemplo: Fac-001 Nro de Control (número mayor que cero y/o cadena de caracteres permitida), Ejemplo: Nc-001 Importe (número mayor que cero), Ejemplo: 2000 Sin derecho a crédito fiscal (número mayor que cero), Ejemplo: 1200 Base imponible (número mayor que cero), Ejemplo: 1000 % Alícuota (número mayor que cero y menor que	Ejemplo: Fac!”·02 Nro de Control (número menor o igual que cero y/o símbolos extraños), Ejemplo: Nc!”·02 Importe (número menor o igual que cero), Ejemplo: -2000 Sin derecho a crédito fiscal (número menor o igual que cero y/o símbolos extraños), Ejemplo: -1200 Base imponible (número menor o igual que cero y/o símbolos extraños), Ejemplo: -1000 % Alícuota			

cien), Ejemplo: 70 Descripción de la cuenta (cadena de caracteres y/o números) y el Código de la cuenta (número de cuenta). Ejemplo: 11102 o Inversiones temporales	(número menor o igual que cero y/o mayor que cien), Ejemplo: -70 Descripción de la cuenta (símbolos extraños) y el Código de la cuenta (cadena de caracteres o símbolos). Ejemplo: ¡”0111			

Caso de Prueba 9

Verificar la funcionalidad de gestionar las notas de crédito y débito.

Verificar la funcionalidad de todas las operaciones correspondientes a la gestión de las notas de débito y crédito.

Caso de uso 9

Gestionar Notas de Crédito y Débito.

Propósito

- Verificar la funcionalidad de todas las operaciones correspondientes a la gestión de las notas de débito y crédito (crear, guardar, confirmar y eliminar una nota si esta se encuentra en estado de edición, modificar y anular una nota si esta se encuentra en

estado de confirmación, buscar las notas de un proveedor, buscar las notas creadas en una fecha determinada, abrir una nota, y asociar una factura a una nota).

Descripción General del caso de uso

- Este caso de uso consiste en Gestionar las Notas de Crédito y Débito.

Flujo Central

- El usuario solicita al sistema iniciar el caso de uso Gestionar Notas de Crédito y Débito.
- El sistema solicita los datos necesarios para visualizar las notas creadas.
- Si el usuario ordena buscar un proveedor debe editar el RIF del proveedor, luego ordena buscar el proveedor y el sistema visualiza los documentos asociados al proveedor.
- El usuario puede buscar la factura por fecha.
- El sistema visualiza las Notas de Crédito y Débito asociadas al proveedor seleccionado.
- El usuario puede realizar varias operaciones como:
- Crear una Nota de Débito o Crédito.
- Abrir una Nota de Débito o Crédito en estado de edición.
- Abrir una Nota de Débito o Crédito en estado de confirmación.
- Asociar una factura a la Nota de Débito o Crédito.
- Guardar la Nota de Débito o Crédito.
- Confirmar la Nota de Débito o Crédito.
- Eliminar una Nota de Crédito o Débito.
- seleccionar una Nota de Crédito o Débito en estado de confirmación.
- Modificar la Nota de Crédito o Débito en estado de confirmación.
- Anular una Nota de Crédito o Débito en estado de confirmación.

Condiciones de ejecución

- El sistema debe tener definido el codificador de los Conceptos de contrapartida.
- El sistema debe tener configurada las reglas contables.

Clases Válidas	Clases Inválidas	Resultado Esperado con datos válidos	Resultado Esperado con datos inválidos	Observación
El usuario inserta el RIF (número hasta ocho cifras). Ejemplo: P-12548993-3	El usuario inserta el RIF (número con más de ocho cifras) Ejemplo: P-12548923393-3.	El sistema busca el proveedor.	El sistema no permite introducir datos de este tipo.	
El usuario crea una Nota, introduciendo los siguientes campos: Nro de la Nota (número mayor que cero y/o cadena de caracteres permitida), Ejemplo: 1500 Nro de Control (número mayor que cero y/o cadena de caracteres permitida),	El usuario crea una Nota, introduciendo los siguientes campos: Nro de Nota (número menor o igual que cero y/o símbolos extraños), Ejemplo: -1500 Nro de Control (número menor o igual que cero y/o símbolos extraños), Ejemplo: Nc!”·-002	El sistema crea la Nota.	El sistema no permite introducir datos de este tipo.	

Ejemplo: Nc-002 Importe (número mayor que cero), Ejemplo: 13000 Sin derecho a crédito fiscal (número mayor que cero), Ejemplo: 150 Base imponible (número mayor que cero), Ejemplo: 1750 % Alícuota (número mayor que cero y menor que cien) Ejemplo: 75 Descripción de la cuenta (cadena de caracteres y/o	Importe (número menor o igual que cero), Ejemplo: -13000 Sin derecho a crédito fiscal (número menor o igual que cero y/o símbolos extraños), Ejemplo: 150 Base imponible (número menor o igual que cero y/o símbolos extraños), Ejemplo: 1750 % Alícuota (número menor o igual que cero y/o mayor que cien), Ejemplo: -75 Descripción de la cuenta (símbolos			

números). Ejemplo: Movimiento de caja	extraños). Ejemplo: Mov!”·			

2.5. Datos de prueba

Definiciones de clases válidas y no válidas

A continuación se muestran los datos válidos que pueden ser utilizados en el módulo Administración Contable, definiéndose también datos inválidos, para comprobar las funcionalidades del sistema.

Gestionar Comprobantes

Interfaz: Gestionar Comprobantes		
Campo	**Datos Válidos**	**Datos Inválidos**
Descripción	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Débito	Números mayores que 0	Números menores o iguales que 0
Crédito	Números mayores que 0	Números menores o iguales que 0

Cerrar Cuentas Nominales

Interfaz: Cuentas Nominales		
Campo	**Datos Válidos**	**Datos Inválidos**
Descripción del comprobante	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos

Definir Fondo de Caja

Interfaz: Fondo de Caja		
Campo	**Datos Válidos**	**Datos Inválidos**
Código	Número mayores que 0	Número menores que 0
Nombre	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos

Registra Egreso de Caja

Interfaz: Egreso de Caja		
Campo	**Datos Válidos**	**Datos Inválidos**
Importe	Números mayores que 0	Números menores que 0
Descripción del movimiento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Importe a retener	Números mayores que 0	Números menores que 0

Gestionar Factura

Interfaz: Gestionar Factura		
Campo	**Datos Válidos**	**Datos Inválidos**
RIF	Número con el siguiente formato: Letra(A-Z) – 8 dígitos - 1dígito	
Número de factura	Cadenas de caracteres que incluyen:	Caracteres o símbolos que no estén entre los definidos

	• A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	
Número de Control	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Importe	Números mayores que 0	Números menores o iguales que 0
Base Imponible	Números mayores que 0	Números menores o iguales que 0
% Alícuota	Números mayores que 0 <=100	Números menores que 0 >100

Gestionar Notas de Débito y Crédito

Interfaz: Nota de Débito y Crédito		
Campo	**Datos Válidos**	**Datos Inválidos**

RIF	Número con el siguiente formato: Letra(A-Z) – 8 dígitos - 1dígito	
Nro de nota	Números >0 y cadenas de caracteres que incluyen: • A – Z • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	
Nro de control	Número mayores que 0	Número menores que 0
Importe	Número mayores que 0	Números menores o iguales que 0
Base Imponible	Números mayores que 0	Números menores o iguales que 0

Configurar Cuentas Bancarias

Interfaz: Cuenta Bancaria		
Campo	**Datos Válidos**	**Datos Inválidos**
Número	Número > 0 de 15 dígitos	Número < 0 de con menos de15 dígitos

Nombre	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Saldo inicial en Banco	Número mayores que 0	Número menores que 0

Registrar Depósito por Clasificar

Interfaz: Depósito por Clasificar		
Campo	**Datos Válidos**	**Datos Inválidos**
Número de documento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Importe	Números mayor que 0	Números menores o igual que 0
Descripción de la	Cadenas de caracteres que incluyen:	

cuenta	• A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	

Registrar otros ingresos

Interfaz: Otros Ingresos		
Campo	**Datos Válidos**	**Datos Inválidos**
Número de documento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Importe	Números mayor que 0	Números menores o igual que 0
Descripción de la cuenta	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios	Caracteres o símbolos que no estén entre los definidos

	• Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	
Saldo en Libro	Números mayor que 0	Números menores o igual que 0

Registrar Pago de Obligaciones

Interfaz: Pago de Obligaciones		
Campo	**Datos Válidos**	**Datos Inválidos**
RIF	Número con el siguiente formato: Letra(A-Z) – 8 dígitos - 1dígito	.
Número de documento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Importe	Números mayor que 0	Números menores o igual que 0
Importe a retener	Números mayor que 0	Números menores o igual que 0
Descripción de la cuenta	Cadenas de caracteres que incluyen:	Caracteres o símbolos que no estén entre los definidos

	• A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	

Registrar Pago al Contado

Interfaz: Pago al Contado		
Campo	**Datos Válidos**	**Datos Inválidos**
RIF	Número con el siguiente formato: Letra(A-Z) – 8 dígitos - 1dígito	.
Número de documento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Importe	Números mayor que 0	Números menores o igual que 0
Importe a retener	Números mayor que 0	Números menores o igual

		que 0
Descripción de la cuenta	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos

Registrar Reembolso de Caja

Interfaz: Reembolso de Caja		
Campo	**Datos Válidos**	**Datos Inválidos**
Número de documento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos.
Importe	Números mayor que 0	Números menores o igual que 0
Descripción del movimiento	Cadenas de caracteres que incluyen: • A – Z	Caracteres o símbolos que no estén entre los definidos

	• Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	

Registrar Transferencia a Oficinas

Interfaz: Transferencia a Oficinas		
Campo	**Datos Válidos**	**Datos Inválidos**
Número de documento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos.
Importe	Números mayor que 0	Números menores o igual que 0
Descripción del movimiento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: ·	Caracteres o símbolos que no estén entre los definidos.

	$ % & / ( ) = ¿? * _ ,	

Registrar Otros Egresos

Interfaz: Otros Egresos		
Campo	**Datos Válidos**	**Datos Inválidos**
Número de documento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Importe	Números mayor que 0	Números menores o igual que 0
Descripción del movimiento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos

Crear Estado de Cuentas

Interfaz: Estado de Cuentas		
Campo	**Datos Válidos**	**Datos Inválidos**
Número de documento	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Débito	Números mayor que 0	Números menores o igual que 0
Crédito	Números mayor que 0	Números menores o igual que 0

Gestionar Retenciones

Interfaz: Gestión de Retenciones		
Campo	**Datos Válidos**	**Datos Inválidos**
RIF	Número con el siguiente formato: Letra(A-Z) – 8 dígitos - 1dígito	
Descripción del movimiento	Cadenas de caracteres que incluyen: • A – Z • Números	Caracteres o símbolos que no estén entre los definidos

	• Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	
Número de factura	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos.
Número de control	Cadenas de caracteres que incluyen: • A – Z • Números • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos
Importe total	Números mayor que 0	Números menores o igual que 0
Base imponible	Números mayor que 0	Números menores o igual que 0

Editar Comprobante de Retención

Interfaz: Comprobante de Retención		
Campo	**Datos Válidos**	**Datos Inválidos**
Número de operación	Número hasta 3 dígitos	Número mayores de 3 dígitos y/o cadenas de caracteres
Número de factura	Cadenas de caracteres que incluyen: • A – Z • Números hasta 15 dígitos • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos y/o número > 15 dígitos
Tipo de transacción	Cadenas de caracteres que incluyen: • A – Z • Números hasta 15 dígitos • Espacios • Comienzan con Espacios • Incluye todo tipo de caracteres: · $ % & / ( ) = ¿? * _ ,	Caracteres o símbolos que no estén entre los definidos y/o número > 15 dígitos
Compras sin derecho IVA	Números mayor que 0	Números menores o igual que 0

Base imponible	Números mayor que 0	Números menores o igual que 0
% Alícuota	Números > 0 <=100	Números < 0 >100
IVA retenido	Números mayor que 0	Números menores o igual que 0

2.6. Configuración del entorno de prueba

Propósito

Los módulos Inmobiliario, Mercantil y Administración fueron concebidos como aplicaciones de escritorio y desarrollados sobre la plataforma .Net. En las oficinas mercantiles se encontrará básicamente una aplicación que integra los módulos Mercantil y Administración.

En las oficinas inmobiliarias se encontrará básicamente una aplicación que integra los módulos Inmobiliario y Administración.

El siguiente documento describe como se encuentran distribuidos el software y el hardware cuando se configura una oficina a la hora de realizar las pruebas de software, se muestra una simulación del entorno real en el cual se trabajará con la aplicación, o sea de la oficina, esto ayuda para la realización de pruebas y posteriormente hacer una valoración acerca de los resultados obtenidos de estas pruebas.

Alcance

En la configuración del ambiente de prueba se consideran aspectos como:

Los requisitos básicos del hardware; por ejemplo, procesadores, capacidad de memoria, capacidad de disco duro

Requisitos básicos del software; por ejemplo, sistema operativo, herramientas básicas para la producción.

Hardware adicional en este caso periféricos; por ejemplo, escáner e impresora, y además los

drivers que son software necesarios para el debido funcionamiento de estos dispositivos.
Sistema mínimo de herramientas de software necesarias para facilitar las pruebas, la evaluación, y actividades de diagnóstico.

Descripción física de la configuración de una oficina de Inmobiliario

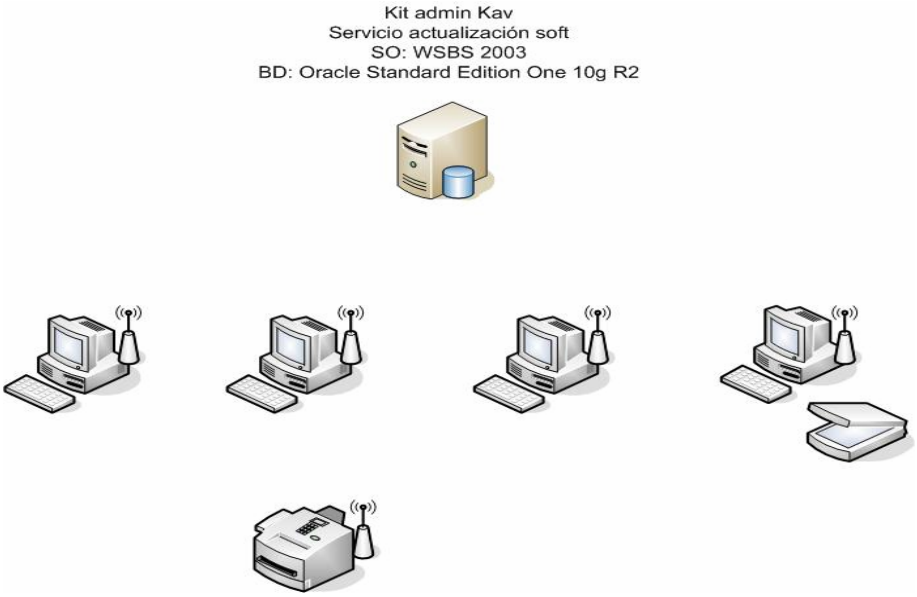

Figura 1. Configuración de una oficina de Inmobiliario.

Descripción breve del entorno de prueba

Para realizar pruebas de software al módulo Administración Contable en una oficina de Inmobiliario es necesario cumplir con las siguientes condiciones:
Para la ejecución estable del software se requiere entre 128 Mb para las estaciones simples, y hasta 512 Mb de RAM para las estaciones de captura y de impresión, dada la función a la que están dedicadas que lleva asociada un número de periféricos que consumen memoria.
Es necesario que exista en cada oficina al menos una impresora y un escáner (conexión por puerto USB 2.0), para realizar operaciones relacionadas con la impresión y escaneo de documentos.
El servidor local de la oficina poseerá el Oracle Standard Edition ONE versión 10g R2 como

gestor de base de datos, y estará montado en sistema operativo Windows Small Business Server.

Inventario de Hardware

Hardware	**Comentario**
Ordenador con 250 GB de disco duro y 2 G de RAM.	Servidor Local
HP LaserJet 2420.	Impresora
HP Scanjet 5590.	Escáner
Modelo del Sistema: HP Compaq dc5100 SFF(PM215AV) Procesador : Intel(R) Pentium(R) 4 CPU 3.00GHz (2 CPUs) Memoria : 512 MB RAM	Ordenador de oficina

Inventario de Software

Software	**Comentario**
Microsoft Windows XP Professional Version 2002	Sistema operativo para ordenadores de oficinas.

Microsoft Windows Small Business Server 2003	Sistema operativo para servidor local.
Oracle Standard Edition ONE versión 10g R2	Gestor de base de datos para servidor local.
Kaspersky Anti-Virus for Window WorkStation 5.0	Antivirus para ordenadores de oficinas
Microsoft Office 2003 SP2	
	Driver de Impresora HP
	Driver de Escaner HP
Framework .Net	Conjunto de lenguajes, herramientas y servicios.

CAPÍTULO 3: EVALUACIÓN DE LA SOLUCIÓN PROPUESTA

3.1. Introducción

En el presente capítulo se desarrollan las pruebas según la solución propuesta o sea se ejecutaron los casos de pruebas diseñados, se aplicó la técnica de prueba establecida en los artefactos plan de prueba y estrategia de prueba. Y se realiza una evaluación de las pruebas.

Las pruebas realizadas están enmarcadas en la etapa final de construcción del software, estos resultados serán entregados al jefe del módulo Administración Contable, para que estas no conformidades sean resueltas. Una vez que esto se haya realizado se ejecutarán nuevamente las pruebas con el objetivo de verificar si el software está listo para pasar al próximo y último nivel de pruebas, que es el de Aceptación.

Los resultados de las pruebas fueron comprobados según los requisitos funcionales. Se aplicó la técnica descrita en la estrategia de pruebas.

Todos los casos de pruebas diseñados se probaron, donde se detectó errores en ocho de ellos. Los resultados de las pruebas han sido registrados y se muestran en las características de la evaluación de las pruebas.

3.2. Evaluación de las pruebas

La evaluación de pruebas recoge, organiza, y presenta los resultados de pruebas obtenidos. Se evalúa la cobertura de los casos de prueba, se analizan los defectos encontrados.

Características de los resultados

Propiedades	**Breve descripción**
Nombre de los caso de pruebas	• Verificar la creación de un comprobante contable manual. • Verificar el cierre de las cuentas nominales.

	• Verificar la selección de fechas para un nivel de cuenta contable determinado. • Verificar según la selección de fechas el nivel de Cuentas Contables. • Verificar que se cree el fondo de caja. • Verificar la funcionalidad de gestionar las notas de crédito y débito. • Verificar la funcionalidad de registrar la asignación de presupuesto. • Verificar la funcionalidad de registrar créditos adicionales. • Verificar la funcionalidad de registrar otros ingresos. • Verificar la funcionalidad de registrar pago al contado. • Verificar la funcionalidad de Registrar Otros Egresos.

	• Verificar la funcionalidad de Crear Estados de Cuentas. • Verificar la funcionalidad de gestionar retenciones. • Verificar la funcionalidad de editar comprobante de retención.
Resultado de las pruebas	Cuando se ejecutaron las pruebas pertenecientes a dichos casos de prueba se comprobó la correcta funcionalidad de los requisitos pertenecientes a cada caso de uso correspondiente a estos casos de pruebas. No se detectó error alguno.
Propósito	Los resultados de las pruebas realizadas representan una validación del caso de uso, este funciona correctamente como se esperaba, realiza todas las acciones correspondientes a los requisitos del cliente. Por lo que este tipo de resultado es sumamente importante.

3.2.1 Clasificación de los defectos

Los defectos encontrados fueron clasificados según tipo y prioridad que presentan, y fueron registrados en un documento entregado al jefe del módulo Administración Contable, para que sean arreglados.

Cada defecto encontrado presenta una prioridad, esta representa la importancia del mismo y la necesidad de ser corregido lo antes posible. Se establecen los siguientes niveles de prioridad:

1. Alto (se definen para aquellos defectos cuya importancia es máxima).
2. Medio (se definen para aquellos defectos cuya importancia es media).
3. Bajo (se definen para aquellos defectos cuya importancia es mínima).

Propiedades	**Breve descripción**	**Elemento**	**Nivel de prioridad**
Nombre del caso de prueba	Verificar los egresos de caja.	Aplicación	
Resultado de la prueba	Al ejecutar la prueba se produjo un error cuando se selecciona cualquier documento, la aplicación muestra un mensaje de error y luego se puede continuar con la operación. Esto no debe ocurrir ya que el		Alto

	mensaje muestra la información de que el usuario que está realizando la operación no tiene los permisos suficientes como para continuar realizando esta funcionalidad. Ver anexo 1.		

Propiedades	**Breve descripción**	**Elemento**	**Nivel de prioridad**
Nombre del caso de prueba	Verificar la funcionalidad de gestionar factura.	Aplicación	
Resultado de la prueba	Al ejecutar la prueba se mostró un mensaje de error cuando se seleccionó un proveedor, la información no es la correspondiente porque si, se puede		Medio

	realizar otra operación, por ejemplo la de Abrir. Ver anexo 2.		

Propiedades	**Breve descripción**	**Elemento**	**Nivel de prioridad**
Nombre del caso de prueba	Verificar la funcionalidad de todas las operaciones correspondientes a la configuración de las cuentas bancarias.	Aplicación	
Resultado de la prueba	El sistema permite crear una cuenta cuyo número sea 0. No se puede crear una cuenta cuyo número sea 0. Cuando se quiere crear otra cuenta bancaria con una cuenta contable (y esta fue utilizada para otra cuenta bancaria) el		Alto

	sistema muestra un mensaje de error, donde informa que ya esa cuenta contable fue utilizada. En realidad es un mensaje de alerta. Ver anexo 3.		

Propiedades	**Breve descripción**	**Elemento**	**Nivel de prioridad**
Nombre del caso de prueba	Verificar la funcionalidad de registrar depósitos por clasificar.	Aplicación	
Resultado de la prueba	El sistema no permite realizar alguna operación sobre los documentos que estén confirmados. Ver anexo 4. Además de no permitir realizar alguna operación sobre los documentos		Alto

	que estén guardados, y el mensaje no explica por qué no se puede efectuar la operación. Ver anexo 5		

Propiedades	**Breve descripción**	**Elemento**	**Nivel de prioridad**
Nombre del caso de prueba	Verificar la funcionalidad de registrar reembolso de caja.	Documento de especificaciones de CU.	
Resultado de la prueba	No se entiende el texto del requisito, ni como probar su funcionalidad.		Alto

Propiedades	**Breve descripción**	**Elemento**	**Nivel de prioridad**
Nombre del caso de prueba	Verificar la funcionalidad de registrar transferencia a oficinas.	1- Documento de especificaciones de CU.	

		2- Aplicación.	
Resultado de la prueba	1.1- En la documentación no se explica como verificar este requisito. 2.1- La Aplicación permite introducir en el campo Importe números negativos. Esto no puede ocurrir. 2.2- No se pudieron probar las funcionalidades de cancelar, modificar y confirmar documentos, debido a que la aplicación no las permite realizar.		Alto Medio Alto

Propiedades	**Breve descripción**	**Elemento**	**Nivel de prioridad**
Nombre del caso de prueba	Verificar la funcionalidad de registrar pago de obligaciones.	1- Documento de especificaciones de CU. 2- Manual de Usuario	

		3- Aplicación.	
Resultado de la prueba	1.1- Existen Condiciones de ejecución que no se cumplen, por lo que no se pudo probar los requisitos. 2.1- En el Manual de Usuario se informa que cuando se accede a la interfaz “Operaciones de Pago de Obligaciones” y se inserta el RIF incorrecto se debe mostrar una E de error y en la Aplicación esto no ocurre. 3.1- Cuando se selecciona un documento de pago en Edición para modificarlo, en la siguiente interfaz sin especificar en Acción a Realizar se cambiaron todos los datos excepto Número Documento, los cambios fueron		Alto Medio Alto

	guardados correctamente. Cuando se cambió el número del documento y se presiona el botón guardar se muestra un mensaje informando que la operación se efectuó satisfactoriamente, pero en realidad la aplicación no cambia el número del documento.		

Propiedades	**Breve descripción**	**Elemento**	**Nivel de prioridad**
Nombre del caso de prueba	Verificar según la selección de fechas el nivel de las cuentas de ingreso y de egreso.	Aplicación.	
Resultado de la prueba	El botón cerrar del menú superior de la pantalla reporte no se activa. Para que esto suceda se debe presionar el botón Actualizar.		Bajo

3.2.2 Cobertura de los casos de pruebas

De un total de veintidós casos de prueba diseñados, ocho de ellos presentaron errores.

El resultado de la cobertura de las pruebas realizadas es el siguiente:

- Casos de prueba realizados = 22 / 95= 23.15 %
- Casos de prueba correctos = 14 / 22 = 63.63 %

Análisis de los defectos encontrados

En algunos casos de pruebas cuando fueron ejecutados se detectaron varios errores en la aplicación y en la documentación, como se muestra en las tablas anteriores. Estos defectos se enumeran de la siguiente manera: ocho de nivel Alto, cinco de nivel Medio y uno de nivel Bajo.

Análisis de los resultados

Los resultados de las pruebas necesitan ser determinados siempre que ocurra la prueba y la ejecución de la evaluación. Esta última puede ocurrir muchas veces durante el ciclo de vida del desarrollo, los resultados de estas deben ser determinados y almacenados de una manera tal que puedan ser repasados y evaluados individualmente para cada caso de ejecución de prueba.

El módulo Administración Contable ha sido probado durante todo su ciclo de vida, los resultados de las versiones anteriores han sido registrados y entregados al jefe de módulo y a su vez estos han sido corregidos.

La última versión de las pruebas realizadas al módulo, no muestran prácticamente errores debido a que este software se está probando hace más de un año y ya está en la etapa final de construcción, y sus fallos se han corregido según se detectaron.

Se concluye que con la planificación de las pruebas, con la información obtenida de los elementos de software a ser probados y de los requisitos de alto nivel, se obtuvo buenos resultados en cuanto a la calidad y eficiencia de las pruebas ejecutadas.

Conclusiones Generales

La realización de este Trabajo de Diploma ha aportado importantes conocimientos a su autor, con el desarrollo del proceso de prueba. Para llevar a fin el objetivo principal se estudiaron las últimas tendencias y tecnologías actuales de las pruebas.

Se identificaron los artefactos necesarios, para ejecutar las pruebas al módulo Administración Contable de los Registros y Notarías de la República Bolivariana de Venezuela y detectar el mayor número de errores existentes, basado en la situación problémica actual.

Llegando así a obtener un adecuado proceso de prueba para el módulo Administración Contable de los Registros y Notarias de la República Bolivariana de Venezuela, lográndose obtener buenos resultados en cuanto a la calidad y eficiencia de las pruebas efectuadas.

Finalmente la investigación ha servido para consolidar conocimientos en esta rama de la ingeniería informática.

Recomendaciones

Se recomienda para las futuras versiones del proyecto Registros y Notarías de la República Bolivariana de Venezuela, emplear las herramientas propuestas por RUP como apoyo de trabajo al equipo encargado de desarrollar las pruebas de software. Y que se recurra al método de automatización de pruebas, para consumir menos esfuerzo y tiempo que el empleado en las pruebas manuales, y garantizar así un despliegue más rápido del software.

Bibliografía

1. **PRESSMAN, Roger.** *"Ingeniería del Software. Un enfoque práctico". .* España : IEEE, 2002.

2. **Mañas, Jose A.** Prueba de Programas. [En línea] 16 de Marzo de 1994.

3. **Departamento de Ingeniería de Software, UCI.** Flujo de trabajo de pruebas. Habana : s.n., 2006.

4. **Rioja, Arturo Mora.** PRUEBA DE APLICACIONES. 2005.

5. —. PRUEBA DE APLICACIONES. *Pruebas Unitarias.* 2005.

6. *Todo Programación.* **Studio Press, S.L.** Madrid, Spain : s.n., 2005.

7. **Mañas, Jose A.** Prueba de Programas. *Pruebas de integración.* [En línea] 16 de Marzo de 1994.

8. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.** Mejora de la calidad de los requisitos mediante la. *Pruebas de Sistema.*

9. **Mañas, Jose A.** Prueba de Programas. *Pruebas de Aceptación .* [En línea] 16 de Marzo de 1994.

10. —. Prueba de Programas. *Ttipos de pruebas.* [En línea] 16 de Marzo de 1994 .

11. **Jordi Mas, David Megías Jiménez, Marc Gibert Ginestà, Álvaro Peña González.** Ingenieria de software en entornos de SL. España : Eureca Media, SL, 2005.

12. **Ceballos, Geyquel Raul Moreno.** SERVICIO AUTÓNOMO DE REGISTROS Y DEL NOTARIADO: MÓDULO INMOBILIARIO. *Desarrollo basado en pruebas.* 2006.

13. **Lanvin, Daniel Fernández.** Desarrollo de una metodología para un nuevo paradigma de desarrollo de software. 2004.

14. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* La Habana : Félix Varela, 2004.

15. **Acuña, César Javier.** Pruebas de Software. s.l. : Universidad Rey Juan Carlos, 2006.

16. **Rational, Corporación de Software.** Plan de Pruebas. *Ayuda de la Herramienta Case Rational.* 2003.

17. —. Estrategia de Pruebas. *Ayuda de la Herramienta Case Rational.* 2003.

18. **Usaola, Macario Polo.** Pruebas de programas Java mediante JUnit. [En línea] Escuela Superior De Informática Universidad De Castilla-la Mancha, 2007.

19. **Popescu, Alexandru.** TestNG. [En línea] 27 de Abril de 2004.

20. **Murciano, Raul.** Comparativa sobre frameworks de pruebas. [En línea] 27 de Julio de 2005 .

21. **weblog, Jason E. Sweat´s.** Simple Test, Unit testing for PHP. [En línea] 2007.

22. **Software, Rational.** IBM Rational Functional Tester. 2007.

23. **gfernandez, Giovanny Fernandez.** Cómo utilizar NUnit. [En línea] 27 de Agosto de 2004.

24. **Rueda, Ana López-Mancisidor.** Solución de IBM Rational paramejorar la calidad funcional de aplicaciones software. 2006.

25. **Hill, Benjamín.** QA Wizard. *Automatización de Pruebas - QA Wizard.* [En línea] 2005.

26. **Corporation, Microsoft.** Microsoft Developer Network. [En línea] Microsoft Corporation. http://msdn2.microsoft.com/es-es/library/ms182516(VS.80).aspx.

GLOSARIO DE TÉRMINOS

A

ASCII

El código ASCII (acrónimo inglés de American Standard Code for Information Interchange — Código Estadounidense Estándar para el Intercambio de Información), pronunciado generalmente [áski], es un código de caracteres basado en el alfabeto latino tal como se usa en inglés moderno y en otras lenguas occidentales. Utiliza 7 bits para representar los caracteres.

C

Casos de uso

Es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.

D

Defecto Software

Se puede definir como una falla o imperfección en un Producto de Software o un Proceso de Software.

E

Eclipse

Es una plataforma de software de código abierto. Esta plataforma, típicamente ha sido usada para desarrollar entornos integrados de desarrollo (del Inglés IDE), como el IDE de Java llamado

Java Development Toolkit (JDT) y el compilador (ECJ) que se embarca como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Error de software

Un error de software o computer bug, es el resultado de una falla de programación introducida en el proceso de creación de programas de computadora.

F

Factura

Comprobante de venta; cuenta detallada que el vendedor entrega al comprador y que muestra todos los detalles de la venta.

Framework

Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

J

JDK

El Java Development Kit, JDK por sus siglas en ingles, es un compilador y conjunto de herramientas de desarrollo para la creación de programas independientes y applets java.

K

Kent Beck

Es uno de los creadores de la metodología ágil para el desarrollo de software conocida como programación extrema. También creó, junto con Erich Gamma el framework de pruebas unitarias para Java, JUnit.

L

Línea base

Concepto de gestión de la configuración del SW. Una especificación o producto que se ha revisado formalmente y sobre los que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambios.

R

Rif

El Rif (Registro de Información Fiscal) es una identificación tributaria de todas las entidades jurídicas existentes.

Requerimiento

Funcionalidad requerida por un usuario para resolver un problema o satisfacer uno o varios objetivos.

S

Saldo Crédito

El saldo de una cuenta en la cual el valor total de los créditos excede el valor total de los débitos.

Saldo Débito

El saldo de una cuenta en la cual el saldo total de los débitos excede el valor total de los créditos.

V

Visual SourceSafe

Es un sistema de control de versiones en el nivel de archivos, que permite a muchos tipos de organizaciones trabajar en distintas versiones de un proyecto al mismo tiempo.

W

Ward Cunningham

Es un programador de "patterns" (patrones para programación") y del WikiWikiWeb.