UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 7

PROCEDIMIENTO GENERAL DE PRUEBAS DE CAJA BLANCA APLICANDO LA TÉCNICA DEL CAMINO BÁSICO

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Autoras: Yaimí Márquez Alpízar
Yenni Valdés Hechavarría

Tutoras: Ing. Regla María Silva Calderón
Ing. Katia Hurtado Duvergel

Asesora: Msc. Martha Rosa Abreu Bosh

Ciudad de La Habana, junio de 2007

DECLARACIÓN DE AUTORÍA

Declaramos ser autoras de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ***25*** días del mes de ***junio*** del año ***2007***.

Autoras:

Yaimí Márquez Alpízar	**Yenni Valdés Hechavarría**

Tutoras:

Ing. Regla María Silva Calderón	**Ing. Katia Hurtado Duvergel**

Agradecimientos:

*A **Reglita** por darle forma a este trabajo, por siempre darnos las fuerzas para continuar, por estar dispuesta a ayudar y brindarnos lo mejor de sí a la hora que la necesitamos.*

*A **Pedro Medina** por las lecciones de programación, paciencia y comprensión que nos ofreció en cada momento.*

*A **Katia** por escucharnos y atendernos siempre que se lo pedimos.*

*A **Lourdita** por la incondicionalidad y la franqueza.*

*A **Rosita** por sus sabios consejos y por todas las molestias ocasionadas.*

*A las **megas** por darnos la luz y llevarnos al lugar exacto y en el momento preciso.*

*A la **UCI** por estos 5 años de tristeza y alegría que vivirán por siempre en nuestros corazones.*

A todos Muchas Gracias

Yaimí:

*- A **mis padres**, a los que no alcanzaría toda una vida para agradecerles.*

*-A mi **hermana,** por soportar mis malcriadeces, por guiar mis pasos y mostrarme el camino.*

*- A **mi novio** gracias por existir y por ser tan maravillosamente especial.*

*- A **mis abuelos Vice y Raude** por siempre preocuparse y tenerme presente.*

*- A **Yuni y Amaury** por todo el apoyo y el cariño y por estar siempre ahí cada vez que los necesité.*

*- A **Yely** por darle presencia a este trabajo.*

Yenni:

*- A **mis padres** por haberme dado la vida y saber guiarme en todo momento.*

*- A **mi queridísimo hermano** por darme la gran felicidad de ser tía.*

*- A **Rey** por darme todo el amor y las fuerzas para terminar mis estudios.*

*- A **Sara, Puerta, Rolando, Are y Mire** por siempre preocuparse por mí y ofrecerme su mano amiga en cualquier circunstancia de la vida.*

A nuestros padres, con todo el amor del mundo.

Yaimí y Yenni

Resumen

Uno de los mayores problemas que se afronta actualmente en la esfera de la informática es la calidad del software. El proceso de pruebas es sin dudas unos de los aspectos fundamentales para medir el estado de calidad de un sistema informático. Este trabajo tiene como objetivo proponer un Procedimiento para la realización de las pruebas de caja blanca usando la técnica del camino básico a los software producidos en la Facultad 7 de la Universidad de las Ciencias Informáticas.

En esta investigación se hizo un analisis de las principales bibliografías especializadas en el tema, profundizando en los diferentes métodos de pruebas que existen, fundamentalmente todas las técnicas encaminadas a la revisión del código fuente de un sistema informático. Además se analizó la situación existente actualmente en la universidad con respecto a la calidad de software y a la realización de las pruebas. Todo ello permitió estandarizar el procedimiento según lo establecido por la Norma ISO 9000 del 2005.

Si se generaliza la aplicación de este procedimiento en la revisión de los software de la facultad 7, el proceso de control de calidad alcanzará un nivel mucho más elevado y se facilitará el trabajo de los probadores que lo utilicen. Para la evaluación de este procedimiento se implantó el mismo en la revisión del código fuente del Módulo Dmail del Sistema Cassandra PACS elaborado por el Grupo de Procesamiento de Imágenes y Señales de la facultad obteniéndose resultados satisfactorios con su aplicación.

ÍNDICE

ÍNDICE DE FIGURAS

Introducción

Las ciencias informáticas y paralelamente la producción de Software, han alcanzado en la actualidad un elevado auge e importancia a nivel mundial. Su desarrollo crece de forma vertiginosa y con ello la demanda de mejores software con menores tiempos y costos de producción. En Cuba también se han notado avances en este sentido, ya que la vinculación de todas sus ramas: económicas, políticas y sociales con el mundo Informático son de primordial interés para el Estado Cubano, no sólo por los beneficios que trae desde el punto de vista del desarrollo de sistemas para el uso interno, sino también con el objetivo de introducirse en el mercado a escala mundial aprovechando su perspectiva económica.

Surgida al calor de la Batalla de Ideas la Universidad de las Ciencias Informáticas avanza con mucha firmeza en la industria del software, lo que inicialmente pudo ser una idea de producir programas informáticos se ha convertido en un proceso bastante profundo y cada vez más complejo.

Por esta razón ya se cuenta en la UCI con un Laboratorio para la Certificación de Calidad de los software de producción interna llamado ***Calisoft***, y aunque mucho falta por lograr en estos aspectos el trabajo realizado hasta el momento ha sido bastante intenso.

"*La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del mismo que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software*"¨. [Pressman, 2000].

"*En el desarrollo de software, las personas involucradas, cometen errores, suelen equivocarse en algunas formas características, y según algunos autores, hay cierta tasa de errores que es estadísticamente predecible*". [cig_labs, 2002].

Es por ello que si se le aplican las pruebas requeridas a los productos en su etapa de desarrollo entonces se logra que salgan al mercado con un mínimo de errores y que tengan una buena

aceptación por parte de los clientes. No debe dejarse para el final la aplicación de las pruebas como se acostumbra a hacer ya que si se hacen durante todas las etapas de desarrollo del mismo se evitará que se cometan errores que después serian bastante graves e incluso en algunas ocasiones nunca podrían llegar a ser descubiertos.

El equipo de **Calisoft** antes mencionado además de la dirección central cuenta con un grupo de apoyo en cada una de las diez facultades de la Universidad a los cuales se asignan diversas tareas relacionadas con la calidad, entre ellas las pruebas a los productos de software terminados. Estos equipos sirven de guía y controlan de cierta forma todo el proceso de calidad dentro de la facultad correspondiente.

El Equipo perteneciente a la Facultad 7 recibe software terminados y destinados a la salud y al procesamiento de imágenes. A estos sistemas se le realizan pruebas de especificación y de documentación, en las que por la experiencia ya obtenida se tiene al menos una metodología o un plan de prueba a seguir a la hora de realizar las mismas.

Existen otras pruebas que aunque son de vital importancia en el proceso de control de calidad de un sistema informático aún no han sido aplicadas en la Facultad 7 y en los casos de haberlas realizado, nunca con la calidad y profundidad que requieren las mismas por motivos tales como la carencia de un estándar o un procedimiento a seguir, por falta de tiempo o de personal capacitado en los temas.

Tal es el caso de las Pruebas de Código o de Caja Blanca como también se les conoce, las cuales se comienzan a aplicar en la Facultad 7 sin un procedimiento o estándar que facilite el arduo trabajo de realizar las mismas. Por la vital importancia que tienen los software destinados a la salud, ya que cualquier error en los mismos podría costarle hasta la vida a personas innecesariamente es importante que la revisión del código fuente de estos programas se realice de la forma más completa posible y con la calidad requerida.

La no existencia de un procedimiento para la realización de las Pruebas de Caja Blanca así como el escaso conocimiento práctico acerca de este tema, es una cuestión a la que se le debe dar solución

para lograr que la revisión del código de los sistemas informáticos de la Facultad 7 sea un proceso con la calidad y profundidad con que requieren los mismos.

Se puede definir entonces el siguiente **problema científico:** ¿Cómo aplicar las Pruebas de Caja Blanca usando la Técnica del Camino Básico a los productos de software terminados en la Facultad 7 de la Universidad de las Ciencias Informáticas?
Este problema se enmarca en el **objeto de estudio:** Proceso de Pruebas de Software en la Facultad 7.

El **campo de acción** es el Proceso de Pruebas de Caja Blanca usando la Técnica del Camino Básico en la Facultad 7.

El **objetivo general** que se persigue con la realización de este trabajo es: Proponer un procedimiento para la realización de las Pruebas de Caja Blanca usando la Técnica del Camino Básico a los productos de software elaborados en la Facultad 7 de la Universidad de las Ciencias Informáticas.

Para encaminar la investigación en vista a resolver el problema planteado se propone la siguiente **Hipótesis:** si se propone un procedimiento estándar para la realización de las pruebas de caja blanca a los productos elaborados en la Facultad 7 entonces se garantizará una mejor revisión de estos sistemas informáticos.

Las **tareas** que se llevan acabo para darle cumplimiento a los objetivos trazados son:
Fundamentar las pruebas de software especificando sus objetivos, estrategias a seguir a la hora de aplicarlas y cómo se debe hacer su planificación.

1. Fundamentar las pruebas de software especificando sus objetivos, estrategias a seguir a la hora de aplicarlas y cómo se debe hacer su planificación.
2. Profundizar en el estado actual de las pruebas de software en la Facultad 7.
3. Investigar las Técnicas de Pruebas de Caja Blanca que más se utilizan, sus objetivos y pasos.

4. Proponer un procedimiento para la realización de las pruebas de Caja Blanca usando la técnica del Camino Básico.
5. Implantar dicho procedimiento en el Módulo DMail del Sistema Cassandra PACS.
6. Documentar los resultados obtenidos con la implantación del procedimiento y los resultados de las pruebas realizadas.
7. Valorar los resultados obtenidos con la implantación del procedimiento propuesto.

El presente trabajo, está estructurado en tres capítulos:

El primer capítulo contiene los conceptos importantes de pruebas, la relación entre las pruebas de Caja Blanca y la calidad de un software, los objetivos y las estrategias a seguir para aplicarlas así como también los tipos de pruebas existentes y los pasos para su diseño.

En el segundo capítulo se propone un procedimiento dirigido a la realización de las Pruebas de Caja Blanca usando la Técnica del Camino Básico siguiendo el formato según lo establecido por la Norma ISO 9000 del 2005.

En el tercer capítulo se describe el sistema que se prueba, se implanta en el Módulo DMail del Sistema Cassandra PACS el procedimiento propuesto anteriormente y se valoran los resultados obtenidos luego de aplicar el procedimiento y con la realización de las pruebas al software.

Capítulo 1 Fundamentos Teóricos

1.1 Introducción

El ciclo de vida de un producto de software empieza en el momento en que nace la idea de desarrollar el sistema y termina cuando el software por una u otra razón deja de ser usado. Entre estos dos momentos el producto de software pasa por varias fases en las que una de las actividades más importantes a desarrollar son las pruebas; de hecho, se ha establecido formalmente que éstas son fundamentales dentro de cada una de las etapas del proceso de desarrollo de un sistema.

La principal razón es que a partir de ellas se puede asegurar el cumplimiento de criterios mínimos de operabilidad y garantizar la calidad de los productos implementados. A pesar de esto, no es difícil percibir como su importancia se ha subestimado y en ocasiones hasta ignorado, resultando un tema desconocido y/o menospreciado por gran parte de los desarrolladores.

La prueba es un elemento crítico para la garantía de la calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación. Constituyen una etapa dentro del desarrollo de cualquier aplicación y un tema importante dentro de la ingeniería de software.

En este capítulo se presenta el estado del arte de este tema, teniendo en cuenta los objetivos. Son analizados algunos aspectos como: la Metodología RUP y las pruebas en cada una de sus fases, la relación que existe entre las pruebas y la calidad de un software, así como los tipos de pruebas que existen, los objetivos y estrategias a seguir para aplicarlas.

1.2 Metología RUP

RUP es en esencia un proceso o metodología de desarrollo de software: Es una forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo (quién hace qué, cuándo y cómo), y tiene como objetivo asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles. Este proceso está dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

Con esta metodología tiende a aumentar la productividad de los desarrolladores mediante acceso a: base de conocimiento, plantillas y herramientas. Se centra en la producción y mantenimiento de modelos del sistema más que en producir documentos. Es una guía de cómo usar UML de la forma más efectiva.

El proceso de ciclo de vida de RUP se divide en cuatro fases. Esas fases se dividen en iteraciones, cada una de las cuales produce una pieza de software demostrable.

1.2.1 Fases de la Metodología RUP

Incepción: Tiene como propósito establecer la oportunidad y alcance del proyecto y proponer una visión muy general de la arquitectura de software.

Elaboración. Tiene como objetivos realizar el análisis del dominio del problema y definir el plan del proyecto donde se planifiquen las actividades necesarias y recursos requeridos.

Construcción. El propósito es completar la funcionalidad del sistema para ello se deben clarificar los requerimientos pendientes, administrar el cambio de los artefactos construidos, ejecutar el plan de administración de recursos y mejoras en el proceso de desarrollo para el proyecto.

Transición. Comienza cuando el producto está suficientemente maduro para ser entregado. Se corrigen los últimos errores.

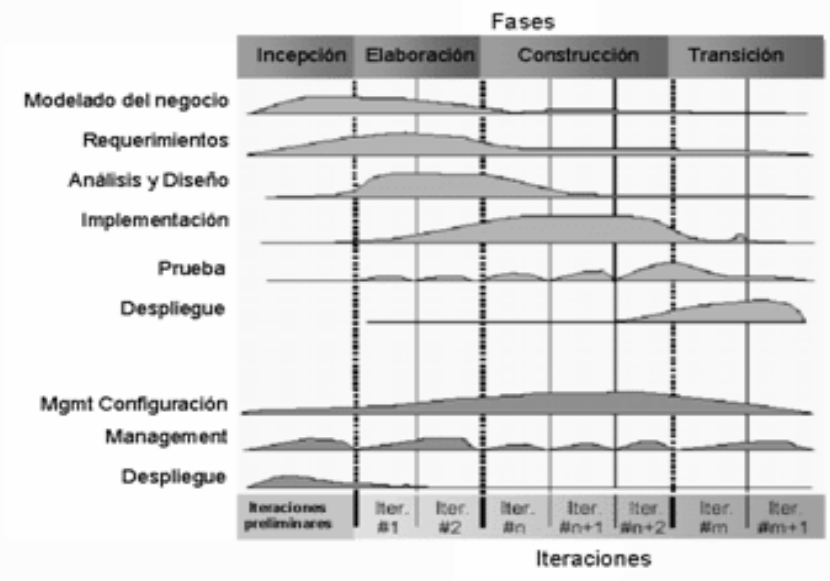


Figura 1. Rup en dos Dimensiones

1.2.2 Flujos de trabajo de la Metología RUP

- ✓ **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.

- ✓ **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

- ✓ **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

- ✓ **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

- ✓ **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.

- ✓ **Instalación:** Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.

- ✓ **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

- ✓ **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización, actualización concurrente de elementos, control de versiones, etc.

- ✓ **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

1.2.3 Pruebas en la Metodología RUP

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

En la Fase de Inicio.

Se comienzan a considerar que pruebas se requerirán y se van desarrollando algunos planes provisionales de prueba. No se realiza en esta etapa un trabajo significativo de pruebas ya que el prototipo exploratorio de demostración tiene por lo general carácter ilustrativo más que operativo. El líder de proyecto puede considerar útil el dedicar un pequeño esfuerzo a pruebas. Se puede generar un modelo de pruebas algo rudimentario en esta fase.

En la Fase de Elaboración.

El objetivo es asegurarse de que los subsistemas de todos los niveles (subsistemas de servicio y subsistemas del diseño) y de todas las capas (desde la capa del sistema hasta las capas específicas de la aplicación) funcionen. Sólo se pueden probar los componentes ejecutables.

Al empezar por las capas más bajas de la arquitectura se prueban los mecanismos de distribución, almacenamiento, recuperación (persistencia) y concurrencias de objetos, así como otros mecanismos de las capas inferiores del sistema. Con esto no sólo se prueba la funcionalidad sino también el rendimiento. Todas las capas no son necesarias de probar sino es necesario probar cómo las capas superiores hacen uso de las inferiores.

Al planificar las pruebas se seleccionan los objetivos que evaluarán la línea base de la arquitectura.
Al diseñar las pruebas se toman como base estos objetivos para identificar los casos de pruebas necesarios y preparará procedimientos de pruebas para comprobar la sucesiva integración de subsistemas hasta completar la línea base.

Al comprobar los componentes se quedará listo para realizar las pruebas de integración. Al ser integrado el sistema, tal y como queda definido por los casos de uso arquitectónicamente significativos se realizan las pruebas de sistema.

En la Fase de Construcción.

En esta fase las pruebas son una actividad fundamental. Al planificar las pruebas se seleccionan los objetivos que comprueben las sucesivas construcciones, y por último el propio sistema.

Al diseñar las pruebas se determina cómo probar los requisitos en el conjunto de construcciones y se preparan casos y procedimientos de pruebas con este fin.
Se realizan las pruebas de integración informando los resultados para tomar las medidas necesarias en casos de errores.

Se realizan también pruebas del sistema al alcanzarse el estado de versión parcial del sistema, informando los resultados para tomar las medidas necesarias en casos de errores. En esta fase se deben evaluar las pruebas a medida que transcurren las pruebas de integración y del sistema comprobando que éstas alcancen los objetivos del plan de pruebas. Si una prueba no alcanza sus objetivos, los casos y procedimientos de pruebas deberán ser modificados para lograrlos.

En la Fase de Transición

Se pueden realizar pruebas Beta (prueba realizadas en organizaciones representativas “clientes beta”) y/o pruebas Alfa (se realizan en la empresa que desarrolla el software; pero fuera de la organización de desarrollo) y/o validaciones por terceros (una empresa especializada en pruebas realiza pruebas de aceptación por encargo del cliente). Se recopilan y analizan los resultados de estas pruebas con el objetivo de llevar a cabo acciones.

En esta fase se buscan pequeñas deficiencias que pasaron desapercibidas durante la fase de construcción y que pueden ser corregidas en el marco de la línea base de la arquitectura existente.

Se comienza planificando el esfuerzo de prueba en cada iteración y describen luego los casos de pruebas necesarios y sus procedimientos de pruebas. Si es posible se crean a continuación los componentes de pruebas para automatizar algunos de los procedimientos de prueba. Todo esto se hace para cada construcción entregada como resultado del flujo de trabajo de implementación.

Con estos casos, procedimientos y componentes de prueba como entrada, se prueba cada construcción y detectan cualquier defecto. Los defectos se usan como realimentación tanto para otros flujos de trabajo (diseño, implementación) como para los ingenieros de pruebas para que lleven a cabo una evaluación sistemática de los resultados de las pruebas.

De forma general la disciplina de Prueba actúa como un proveedor de servicio a las otras disciplinas en muchos aspectos. La prueba está enfocada principalmente en la evaluación y determinación de la calidad del producto.

1.3 Calidad de Software

La calidad de software es un problema actual que afecta tanto a los productores de software como a los clientes. Con el aumento de la informatización a escala mundial la demanda de software crece exponencialmente y los desarrolladores le han brindado poco interés a la calidad de sus productos. Sucede que muchas veces los clientes reciben el software habiéndose violado la etapa de pruebas.

La calidad del software puede definirse de muchas maneras. Una de las más limitadas, conocida como *"calidad pequeña"* define la calidad como la ausencia de defectos [Kan,1995]. Para evaluarla de esta forma se emplean procedimientos estadísticos a partir de las tendencias de aparición de fallas durante la prueba de software.

Existen estándares industriales que marcan aceptabilidad cuando se estima el número de defectos residuales en 0.02 defectos por millar de líneas de código y aún menos. [Yamaura,1998]

"Otros enfoques de calidad consideran diversos factores, entre ellos la confiabilidad" [Meyer, 1997]. Existe una larga tradición de estudio de la confiabilidad que se asocia estadísticamente con el comportamiento del *software*.

Existen varias formas de definir la confiabilidad. En unos casos se considera tiempo de operación y en otros la variedad de usos propuestos. Una definición más reciente, plantea que: "*La confiabilidad es la probabilidad de operación exitosa de un programa dado, en un intervalo de tiempo, en un ambiente específico*". [Chen y Kao, 1997]

Obteniendo la calidad requerida en el software, se logra reducir su número de errores, o eliminarlos completamente, se alcanza una mayor fiabilidad para las funciones que debe realizar el mismo, mayor eficiencia e integridad de los datos así como mayor flexibilidad y reusabilidad.

"La calidad de software es una actividad de protección que se aplica a lo largo de todo el proceso de Ingeniería del Software. Esta engloba los siguientes aspectos:" [Brito y Napal, 2006]

- Un enfoque de gestión de calidad.
- Tecnología de Ingeniería del Software efectiva (métodos y herramientas).
- Revisiones técnicas formales que se aplican durante el proceso del software.
- Una estrategia de prueba multiescala.
- El control de la documentación del software y de los cambios realizados.
- Un procedimiento que asegure un ajuste a los estándares de desarrollo del software.
- Mecanismos de medición y de generación de informes.

"La calidad debe ser especificada, planificada, administrada, medida y certificada. Esto implica una visión integral que arroja la comprobación del software, con el fin de lograr un mayor grado de satisfacción y confianza del cliente hacia la organización productora de software. Constituye entonces las pruebas de los software, tarea de alta prioridad para las empresas productoras. "[Pressman, 2000]

1.4 Pruebas de Software

Unas de las vías más importantes para determinar el estado de la calidad de un producto de software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante técnicas de prueba. El proceso de pruebas, sus objetivos, métodos y técnicas usadas se describen en el plan de prueba.

La prueba es una actividad fundamental en muchos procesos de desarrollo, incluyendo el del software. Así mismo, una prueba de software permite detectar la presencia de errores que pudieran generar salidas o comportamientos inapropiados durante su ejecución. Un concepto más específico dado por algunos desarrolladores de software es que las pruebas son:

"*Cualquier intento de demostrar que el software tiene propiedades por debajo de la calidad requerida*". [Cig_Labs, 2002].

De acuerdo a la IEEE [IEEE, 1991] el concepto de prueba se define como:

"*Una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente*". [Pressman, 1998].

Otro concepto importante a tomar en consideración es el emitido por Pressman en su edición de 1998, que plantea lo siguiente:

"*La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación".* [Pressman, 1998].

Teniendo en cuenta las definiciones anteriores se puede concluir que la prueba de software es una actividad en la cual el sistema es ejecutado bajo condiciones específicas para demostrar que no tiene la madurez necesaria para ser implantado. Dentro de las actividades para obtener un software con la madurez necesaria están:

- **Revisiones**: consiste en que cada integrante del equipo de desarrollo revisa el producto que va generando.
- **Inspecciones**: revisión de cada producto por parte de colegas.
- **Validaciones**: es el cliente quien revisa el producto para decir si cumple con sus necesidades.

Esta definición implica que se considera una prueba exitosa si se demuestran deficiencias en el software. Las fallas pueden ser en el código o en el modelado, en dependencia del tipo de pruebas que se le apliquen al software.

Se distinguen pruebas técnicas y pruebas funcionales. Las pruebas técnicas son la responsabilidad de los ingenieros de software que han desarrollado el producto, pero estos ingenieros nunca deben hacerse cargo de las pruebas funcionales.

En proyectos a gran escala las pruebas funcionales son la responsabilidad de un equipo de pruebas, formado por uno o varios técnicos, un coordinador de pruebas y un gestor de pruebas o de calidad.

1.4.1 Objetivos de las Pruebas de Software

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

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

Los objetivos anteriores cambian la idea que, normalmente, se tiene cuando se plantea que una prueba exitosa es aquella donde no se detectan errores. *"El objetivo principal es diseñar pruebas que sistemáticamente reflejen diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo".* [Pressman, 2000]

Si las pruebas se llevan a cabo con éxito se descubrirán errores en el software, dándole a éste mayor fiabilidad. Es importante tener en cuenta una frase de Pressman: *"La prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software".* [Pressman, 2000]

1.5 Estrategias de Prueba de Software

En el desarrollo de las pruebas se tienen en cuenta un conjunto de estrategias a seguir para lograr la mayor calidad requerida y el cumplimiento de los objetivos. Se agrupan, de forma muy general, en dos grupos: aquellas que se orientan a observar el comportamiento del sistema. Las pruebas orientadas al comportamiento del sistema son llamadas de **Caja Negra**, funcionales o basadas en las especificaciones. Las pruebas enfocadas sobre la estructura son llamadas de **Caja Blanca** (o transparente) y son basadas en la implementación. La mayoría de los autores recomiendan emplear pruebas de ambos grupos, cuando sea posible.

"Una estrategia de prueba de software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a una construcción correcta del software". [Fernández, 2002]

1.5.1 Características generales de las Estrategias de Prueba.

Para aplicarles las pruebas al software se deben seguir un conjunto de estrategias para lograr que estas se hagan en el menor tiempo posible y con la calidad requerida, además de lograr que arrojen los resultados esperados.

Dentro de las características generales de la estrategia de prueba se encuentran. [Pressman, 2000]

1. La prueba comienza en el nivel de módulo y trabaja "hacia fuera", hacia la integración completa del sistema completo.
2. En diferentes puntos es adecuada la utilización de técnicas de prueba distintas.
3. La prueba la lleva a cabo el que desarrolla el software y para grandes proyectos, un grupo de prueba independiente.
4. La prueba y la depuración son actividades diferentes, pero la depuración puede entrar en cualquier estrategia de prueba.

Hay dos estrategias generales para la prueba de software: las estrategias de prueba de código (Caja Blanca) y prueba de especificación (Caja Negra).

1.5.2 Prueba de Código (Caja Blanca).

La prueba de código se basa en examinar la lógica del programa. Para ejecutarlos se desarrollan casos de prueba que produzcan la ejecución de cada posible ruta del programa o módulo, considerándose una ruta como una combinación específica de condiciones manejadas por un programa.

Hay que señalar que no todos los errores de software se pueden descubrir verificando todas las rutas de un programa, hay errores que se descubren al integrar unidades del sistema y pueden existir errores que no tengan relación con el código específicamente.

1.5.3 Prueba de Especificación (Caja Negra).

En esta prueba se examinan las especificaciones que señalan lo que el programa debe hacer y como lo debe llevar a cabo. Para ejecutarlos se desarrollan casos de prueba reales para cada condición o combinación de condiciones y se analizan los resultados que arroja el sistema para cada uno de los casos. En esta estrategia se verifica el programa considerándolo una caja negra. La prueba no se hace en base al código, sino a la interfaz. No importa cubrir todas las rutas dentro del

programa lo importante es probar todas las entradas en sus valores válidos e inválidos y lograr que el sistema tenga una interfaz amigable.

Limitaciones

Lograr una buena cobertura con pruebas de caja negra es un objetivo deseable; pero no suficiente a todos los efectos. Un programa puede pasar con holgura millones de pruebas y sin embargo tener defectos internos que surgen en el momento más inoportuno.

Por ejemplo, una computadora que contenga el virus Viernes-13 puede estar pasando pruebas de caja negra durante años y años. Sólo falla si es viernes y es día 13; pero ¿a quién se le iba a ocurrir hacer esa prueba?

Las pruebas de caja negra nos convencen de que un programa hace lo que queremos; pero no de que haga (además) otras cosas menos aceptables.

1.6 Plan de Prueba

"*El propósito del plan de pruebas es dejar de forma explícita el alcance, el enfoque, los recursos requeridos, el calendario, los responsables y el manejo de riesgos de un proceso de pruebas".* [Teruel, 2001]

Está constituido por un conjunto de pruebas. Cada prueba debe:

- Dejar claro qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad,...).
- Dejar claro cómo se mide el resultado.
- Especificar en qué consiste la prueba (hasta el último detalle de cómo se ejecuta).
- Definir cuál es el resultado que se espera (identificación, tolerancia,...). ¿Cómo se decide que el resultado es acorde con lo esperado?

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

Estas mismas ideas se suelen agrupar diciendo que un caso de prueba consta de 3 bloques de información:

1. El propósito de la prueba.

2. Los pasos de ejecución de la prueba.
3. El resultado que se espera.

Todos y cada uno de esos puntos deben quedar perfectamente documentados. El plan de pruebas señala el enfoque, los recursos y el esquema de actividades de prueba, así como los elementos a probar, las características, las actividades de prueba, el personal responsable y los riesgos.

1.7 El Proceso de Pruebas

El proceso de pruebas de un software consta de varias etapas dentro de ellas las más importantes son:

1. **Inspección del análisis:** Se verifica si se cometieron errores o falla en la etapa de análisis.
2. **Inspección del diseño:** Se comprueba el diseño y se trata de hallarle defectos.
3. **Inspección del código:** Se observa el entendimiento y facilidad del código.
4. **Pruebas unitarias:** Se debe probar cada método de las clases implementadas por separado.
5. **Pruebas de integración:** Se prueban todas las clases, verificando que compaginen entre sí.
6. **Pruebas de validación de requerimientos:** Validan que se cumple con todos los requerimientos exigidos por el cliente.
7. **Pruebas de sistema:** Ejecutar el programa para verificar si cumple con los requisitos exigidos.

1.7.1 Pruebas de Unidad

Se comprueban los módulos cada uno por separado, buscando errores en el funcionamiento de ese módulo como sistema independiente. Estas pruebas deben ser hechas por el diseñador y el programador del módulo.

1.7.2 Pruebas de Sistema

Su objetivo es la comprobación del sistema global, se realizan pruebas de tres tipos distintos:

- **Seguridad:** protección en aplicaciones especialmente sensibles a entradas no deseadas.
- **Resistencia:** se prueba la robustez del sistema frente al mal uso de la aplicación por parte de ciertos usuarios.
- **Rendimiento:** Eficiencia medida en velocidad de proceso y recursos consumidos.

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

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

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

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

1.7.4 Pruebas de Aceptación

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de su puesta en marcha es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada 'prueba de aceptación'. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique.

Para eliminar la influencia de conflictos de intereses, y para que sea lo más objetiva posible, la prueba de aceptación no debería ser responsabilidad de los ingenieros de software que han desarrollado el producto.

Para la preparación, ejecución y evaluación de la prueba de aceptación no se requiere de conocimientos informáticos. Sin embargo, un conocimiento amplio de métodos y técnicas de prueba y de la gestión de la calidad en general facilita esta labor.

La persona adecuada (o el equipo adecuado) para llevar a cabo la prueba de aceptación dispone de estos conocimientos y además es capaz de interpretar los requerimientos especificados por los futuros usuarios del sistema de software en cuestión.

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, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcional pactada previamente con el cliente.

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 comienza a usarlo.

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

Una prueba de aceptación puede ir desde un informal caso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas. De hecho, las pruebas de aceptación pueden tener lugar a lo largo de semanas o meses, descubriendo así errores latentes o escondidos que pueden ir degradando el funcionamiento del sistema. Estas pruebas son muy importantes, ya que definen las nuevas fases del proyecto como el despliegue y mantenimiento.

Se emplean dos técnicas para las pruebas de aceptación:

- **La prueba alfa.**

Se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario. Las pruebas alfa se llevan a cabo en un entorno controlado. Para que tengan validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto, se procede a realizar las pruebas y a documentar los resultados.

- **La prueba beta.**

Se lleva a cabo por los usuarios finales del software y se realiza en el entorno de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. Así, la prueba beta es

una evaluación "en tiempo real" del software en un entorno no planificado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador.

Como resultado de los problemas informados durante la prueba beta, el desarrollador del software lleva a cabo modificaciones y así prepara una versión del producto de software para todos los cliente donde se despliegue el producto.

1.8 Métodos de Pruebas

Existen diversos métodos para realizar las pruebas de software, entre las más importantes se encuentran la Prueba de Caja Blanca, Prueba de Caja Negra y Prueba de la Estructura de Control.

La prueba de Caja Blanca es la mejor de su tipo para verificar que se recorran todos los caminos y detectar un mayor número de errores. La Caja Negra brinda la posibilidad de cubrir la mayor parte de las combinaciones de entradas y lograr con ello un juego de pruebas más eficaz.

Las pruebas mencionadas permiten probar cada una de las condiciones existentes en el programa, identificar claramente las entradas, salidas y estudiar las relaciones que existen entre ellas, permitiendo así maximizar la calidad de las pruebas y en dependencia del resultado se constará con un sistema más estable y confiable.

1.8.1 Las Pruebas de Caja Blanca

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que:

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran

porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. [Pressman, 2000]

1.8.1.1 Prueba del camino básico

La prueba del camino básico es una técnica de prueba de Caja Blanca propuesta por Tom MacCabe. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el **Grafo de Flujo** asociado y se calcula su **complejidad ciclomática.** Por último se diseñan los casos de prueba y se ejecutan los mismos.

Esta técnica ofrece una gran ventaja con respecto a otros de su tipo ya que el número mínimo requerido de pruebas se sabe por adelantado y por tanto el proceso de prueba se puede planear y supervisar en mayor detalle que con la mayoría de las otras estrategias.

1.9 Flujo Actual de los Procesos

El proceso de control de calidad antes, durante y después de la implementación de un producto de software es una tarea bastante complicada incluso para los expertos en el tema, ya que nunca se tiene la última palabra a cerca de estos factores y cada situación que se presenta puede resultar novedosa y problemática simultáneamente.

En la Facultad 7 de la Universidad de las Ciencias Informáticas se producen sistemas destinados a la salud y en su mayoría con dimensiones bastante grandes y compuestos por varios módulos integrados entre si.

El proceso de pruebas a estos sistemas sólo se realiza una vez estén terminados por el mismo personal implementador del producto, o en un laboratorio de calidad que aún no cuenta con una elevada preparación de sus miembros en los temas. Las pruebas que se realizan son de Caja Negra y a la Documentación por lo general, el jefe del equipo de prueba recibe de manos del líder del proyecto todo el sistema y la documentación a probar y a medidas que se realiza la revisión se van conformando los registros de no conformidades, estos últimos los recibe el líder del proyecto para

hacer cambios a favor de superar la calidad de la aplicación y nuevamente someterla a otra revisión y así se repite el proceso hasta que el producto quede con el mínimo número de errores posible.

Las pruebas de caja blanca se están comenzando a realizar en algunos grupos productores de software de la facultad, pero de una forma bastante inestable e informal, no se registran en una plantilla oficial los resultados de la revisión, no se sigue ninguna serie de pasos estandarizados, o sea cada proyecto revisa el código fuente usando la técnica más conveniente o que mejor conozca y los pocos que documentan los resultados de las pruebas no lo hacen debidamente.

1.10 Conclusiones

La calidad del software está estrechamente relacionada con las pruebas que se le realizan al mismo, de aquí la importancia que se le atribuye a todo el proceso de pruebas.

Un elemento importante a tener en cuenta en el proceso de prueba es llevar adecuadamente toda la documentación, así como los elementos a probar, las características, las actividades de prueba, el personal responsable y los riesgos asociado, además de documentar todos los resultados obtenidos con la aplicación de las mismas.

Las pruebas de Caja Blanca son de vital importancia para revisar un software y detectar la mayor cantidad de errores posible para que el sistema finalmente tenga una mayor calidad.

Es importante que los desarrolladores tomen conciencia de la influencia de este proceso en los resultados finales de los productos y la gran significación del mismo para lograr competir en el mercado ya sea nivel nacional o internacional.

Capítulo 2 Procedimiento para la Realización de las Pruebas de Caja Blanca usando la Técnica del Camino Básico

2.1 Introducción

En el control de la calidad de un software es muy importante llevar a cabo la revisión del código fuente mediante las Pruebas de Caja Blanca. Para que estas sean realizadas con la calidad y profundidad requerida hay que seguir una serie de pasos y completar toda la documentación que se genera durante la puesta en marcha de las mismas.

Este procedimiento pretende proporcionar la secuencia de acciones necesarias para lograr el diseño e implantación de los casos de Pruebas de Caja Blanca usando la Técnica del Camino Básico y de esta forma mejorar la situación existente actualmente en la Facultad 7 de la Universidad de las Ciencias Informáticas en cuanto a la revisión del código fuente de los software terminados.

El Formato usado para el diseño de este procedimiento cumple con los estándares definidos en la Norma ISO 9000 del 2005. *(Ver Anexo 1)*

2.2 Objetivo

- ✓ Establecer los principios de trabajo de la Facultad 7 para la ejecución de las Pruebas de Caja Blanca a Productos de Software terminados usando la Técnica del Camino Básico.
- ✓ Especificar cada una de las actividades a seguir durante la realización de estas pruebas así como las plantillas a utilizar en cada caso.
- ✓ Facilitar el trabajo del personal dedicado a la ejecución de las Pruebas de Caja Blanca en la Facultad 7.

2.3 Alcance

- ✓ Este procedimiento abarca aquellos productos de software terminados que se revisan en la facultad 7 y de los cuales se tenga disposición del código fuente, además de la documentación necesaria.

- ✓ El mismo va dirigido a aquellas personas que pertenecen a la facultad 7 y que se encargan de la revisión exhaustiva del código de los software a probar.
- ✓ Estas pruebas no evalúan a los programadores sino al código como tal.

2.4 Definiciones, Acrónimos y Abreviaturas

2.4.1 Documentación del Proyecto

Documentos generados durante la producción del Proyecto por el equipo de desarrollo del mismo y que describen todo el entorno de software y hardware relacionado con este.

2.4.2 Pruebas de Código (Caja Blanca)

La prueba de código se basa en examinar la lógica del programa. Para ejecutarlos el analista desarrolla casos de prueba que produzcan la ejecución de cada posible ruta del programa o módulo, considerándose una ruta como una combinación específica de condiciones manejadas por un programa. [McCabe, 1996]

2.4.3 Caso de Prueba

Instrucciones documentadas para la persona que realiza las pruebas que especifican cómo se tiene o tendría que probar una función o una combinación de funciones. El caso de prueba incluye información detallada sobre los siguientes pasos:

- ✓ Objetivo de la prueba.
- ✓ Funciones que se han de comprobar.
- ✓ Entorno de prueba y otras condiciones (detalles de configuración y trabajos preparatorios).
- ✓ Datos de prueba.
- ✓ Comportamiento esperado del sistema.

2.4.4 Defectos del Producto

Fallas existentes en el producto que hacen que el mismo no cumpla con los requisitos especificados por el cliente para su uso o exista un incumplimiento de los estándares definidos por el Grupo de Aseguramiento de Calidad.

2.4.5 Grafo de Flujo

Se usan para traducir o representar la secuencia de ocurrencia de todo segmento de código de cualquier programa. Un Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y nos brinda información para confirmar que el trabajo se está haciendo adecuadamente: los nodos, las aristas y las regiones. [McCabe, 1996]

2.4.6 Nodo

Cada círculo representado en el grafo se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. [McCabe, 1996]

2.4.7 Arista

Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental. [McCabe, 1996]

2.4.8 Regiones

Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran y la cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa. [McCabe, 1996]

2.4.9 Complejidad Ciclomática

Es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior

para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. [McCabe, 1996]

2.5 Roles y Responsabilidades

2.5.1 Líder de Proyecto (LP)

Actualiza y brinda la información necesaria, las versiones del producto, control de tareas, código generado durante la implementación y documentación derivada de la herramienta de ingeniería de software usada, debidamente modelada con la metodología adecuada y gestionada a través de la herramienta utilizada.

2.5.2 Líder del Equipo de Pruebas (LEP)

Controla el cumplimiento de las normativas para la regularización del proceso de pruebas. Es el encargado de planificar las pruebas al producto, insertarlas dentro del cronograma y mantener siempre la información en orden, para poder brindar los resultados de las pruebas a partir de las No Conformidades detectadas por el Equipo de Revisores. Determina, basándose en análisis realizados sobre lo obtenido, cuando se puede liberar el software para pasar a manos del cliente.

2.5.3 Diseñador de Pruebas

Es el responsable de identificar y definir las pruebas requeridas, y en dependencia de estos tipos de pruebas, asigna a los ingenieros de pruebas para realizar las mismas. En el caso de las pruebas de Caja blanca es el encargado de diseñar todos los casos de prueba y entregárselos al probador con el componente de prueba incluido.

2.5.4 Probador

Es el responsable durante las actividades principales de las pruebas, el cual incluye la conducción de las pruebas necesarias y el registro del resultado de la prueba. Se encarga de ejecutar las pruebas que le entrega el diseñador, no tiene que elaborar el componente de prueba.

2.6 Desarrollo

2.6.1 Introducción a la Técnica del Camino Básico

La técnica del camino básico permite obtener una medida de la complejidad lógica del código de cada método, programa o módulo dado. La idea es derivar casos de prueba a partir de un conjunto

dado de caminos independientes que existen en la codificación por los cuales puede circular el flujo de control. Es además una de las más eficientes en cuanto a cobertura de código, pues logra que se ejecuten todos los bucles en sus límites operacionales.

2.7 Cronograma de Actividades

2.7.1 Recibir el Código Fuente y la Documentación necesaria

- ✓ El líder del equipo de pruebas recibe de manos del líder del proyecto el código fuente del Módulo o sistema al cual se le van a ejecutar las Pruebas de Caja Blanca.

2.7.2 Evaluación de la Documentación

- ✓ El líder del equipo de pruebas en conjunto con otros miembros revisan la documentación entregada.
- ✓ Se analiza si se definió un estándar de código para hacerlo más legible y entendible.
- ✓ Se verifican que estén reflejados todos los requisitos de hardware y software.

2.7.3 Evaluación de los Requisitos de hardware y software

- ✓ Atendiendo a los requisitos especificados en la documentación recogida y a las fechas de entrega del resultado de las pruebas se dispone en el equipo de un determinado número de probadores y de computadoras que cumplan con los requisitos de hardware y software.
- ✓ Se instala un editor de código que reconozca el lenguaje en el que se implementó el sistema que se va a revisar y que si es posible debe coincidir con el mismo que usaron los programadores del software.

2.7.4 Estructuración de las Particiones del Código

- ✓ Aplicar la misma política de partición para todos los módulos del sistema.
- ✓ Se siguen los siguientes criterios de partición:

  a) Por funcionalidad de los métodos o formas: De cada funcionalidad del código, por ejemplo: insertar o eliminar, se divide en fragmentos independientes a probar.

b) Por regiones de código definidas: Si el programador ya definió regiones en el código fuente se hace de cada región una partición a probar.

c) Por caso de uso (si estos no son muy complejos): Cada caso de uso se prueba de forma independiente a través de su código fuente.

✓En cualquiera de los casos se debe seguir el mismo orden lógico del programador y por tanto del programa.

2.7.5 Obtención de los Grafos de Flujo

- ✓ Cada segmento de código de cualquier programa se puede traducir a un *Grafo de Flujo*.
- ✓ Un grafo siempre posee un único nodo inicial y un único nodo final, o sea el grafo nunca puede empezar o terminar en más de un nodo.
- ✓ Si en un segmento de código se tiene una sentencia de la cual se obtiene dos salidas entonces, cada se denominan *nodo predicado* y están caracterizados porque dos o más aristas emergen de él. (Ver Anexo 4).
- ✓ Para construir el grafo se debe tener en cuenta la notación para las diferentes instrucciones. (Ver Anexo 2).
- ✓ Se numeran las instrucciones siguiendo estas notaciones y atendiendo a las particularidades del código fuente.
- ✓ Se diseña el grafo siguiendo la numeración anteriormente establecida.

2.7.6 Cálculo de la Complejidad Ciclomática

La Complejidad ciclomática proporciona una medición cuantitativa de la complejidad lógica de un programa.

Existen tres vías para su cálculo:

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

2. La complejidad ciclomática, *V (G)*, se define como: *V (G) = A- N+ 2* donde: *A* es el número de aristas del grafo y *N* es el número de nodos.

3. La complejidad ciclomática, *V (G)*, también se define como: *V (G) = P* + 1 donde: *P* es el número de nodos predicado contenidos en el grafo *G*.

Se recomienda hacer el cálculo por más de una vía para evitar equivocaciones.

2.7.7 Obtención de los Caminos Básico

Para determinar los caminos básicos se siguen los siguientes pasos:

- ✓ Se usa el valor calculado como complejidad ciclomática ya que define el número de caminos independientes y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

- ✓ Se toma como un camino independiente a todos lo camino del programa que introducen al menos un nuevo conjunto de sentencias de procesamiento o una nueva condición.

- ✓ A partir del grafo de flujo se obtienen los caminos independientes siguiendo las aristas del grafo desde el nodo inicial hasta el final.

2.7.8 Derivación de los Casos de Prueba

- ✓ Determinar los casos de pruebas a partir de los caminos independientes previamente obtenidos.

- ✓ Obtener las entradas del caso de prueba a partir del código fuente.

- ✓ Establecer los resultados esperados con la realización de cada caso de prueba.

- ✓ Conformar la Especificación del Caso de Prueba (Ver Anexo 5).

2.7.9 Confección de la Bitácora de Prueba

- ✓ Es un solo documento que incluye los resultados específicos de todos los módulos probados y los resultados generales durante las pruebas de Caja Blanca al software. *(Ver Anexo 7).*
- ✓ En el caso de ser un solo módulo la bitácora estará conformada solamente con los resultados de este. *(Ver Anexo 6).*
- ✓ Para conformarla se tiene que haber terminado la ejecución de todos los casos de prueba. *(Ver Anexo 6).*
- ✓ Deben estar presente el líder del equipo diseñador, el diseñador de prueba y los probadores que participaron activamente durante el proceso.
- ✓ Se define el ambiente de ejecución que son las condiciones de hardware y software bajo las cuales se realizaron las pruebas. *(Ver Anexo 6).*
- ✓ Dentro de la Bitácora se conforma una tabla de corridas para cada Módulo que se probó. *(Ver Anexo 6).*

2.7.10 Resumen de errores

- ✓ Se confecciona un resumen de errores por módulo y uno de manera general.

2.7.11 Entrega al cliente

Una vez culminda la revisión del código utilizando la técnica del camino básico:

- ✓ Se cita al líder del proyecto
- ✓ El equipo de prueba se reúne y le informa las conclusiones generales del proceso de pruebas.
- ✓ Se le entrega la Bitácora de Pruebas.
- ✓ Se hace entrega además de toda la documentación y el sistema entregado para la realización de las pruebas.
- ✓ Se firma el Acta de Evaluación (Ver Anexo 7)
- ✓ El líder del proyecto analiza los errores detectados y en caso de ser necesario pide una nueva revisión al líder del equipo de pruebas

2.8 Distribución y Archivo

Se confeccionará un original y varias copias de ambas plantillas en el servidor de pruebas y se le entregará una copia además al líder del proyecto. Todo esto de hará en formato digital, en caso de ser posible se debe guardar además una copia impresa. Estos documentos sólo pueden ser modificados por el personal autorizado del Equipo de Pruebas.

2.9 Documentos de referencia

- ✓ "Manual de procedimientos para el desarrollo de proyectos productivos" *v 1.0. UCI.2007*.
- ✓ "Plantilla Especificación de Casos de Prueba" *DD-15.00. Softel.2006.*
- ✓ Senlle A. and Vilar J 2000"ISO 9000 En empresas de servicio" *Ediciones Gestión 2000.*

2.10 Modelos

- ✓ "Plantilla para Especificación de Casos de Prueba de Caja Blanca" v 1.0. Facultad 7.UCI.2007. (Ver Anexo 5)
- ✓ "Bitácora de Pruebas de Caja Blanca" v 1.0. Facultad 7.UCI.2007. (Ver Anexo 6)
- ✓ "Acta de Evaluación" v 1.0. Facultad 7.UCI.2007(Ver Anexo 7)

2.11 Conclusiones

En este capítulo se establece el procedimiento que proporciona la secuencia de acciones necesarias para lograr el diseño e implantación de los casos de las pruebas de caja blanca usando la técnica del camino básico en un formato que cumple con los estándares definidos en la Norma ISO 9000 del 2005.

Capítulo 3 Evaluación del Procedimiento propuesto en el Módulo DMail del Sistema Cassandra PACS.

3.1 Introducción

Con la finalidad de demostrar la hipótesis expuesta se procede a probar el Código Fuente del Módulo Dmail del Sistema Cassandra PACS con la Técnica del Camino Básico de las Pruebas de Caja Blanca, aplicando y evaluando el Procedimiento Propuesto con anterioridad.

En este capítulo se hace una descripción del sistema que se prueba, de las condiciones de hardware y software que se requieren para llevar a cabo las pruebas, se realiza además la evaluación del procedimiento propuesto y los resultados obtenidos con su aplicación.

3.2 Encuesta realizada a los implemetadores del Sistema cassandra PACS

Con el objetivo de conocer y profundizar en el estado de la calidad y de las pruebas de software en la Universidad de las Ciencias Informáticas, se seleccionó el Grupo de Procesamiento de Imágenes (GPI), como parte importante del proceso de inmersión de la UCI en la informática a nivel internacional y a su vez, en el mercado mundial de software, a 30 miembros de este grupo, sobre todo los que han estado directamente vinculados durante más tiempo a la producción se le realizó una encuesta (ver Anexo 8), con vistas a recoger su opinión, y para, de esta manera, tener una muestra del estado del arte a través del conocimiento de los mismos acerca del tema.

La encuesta arrojó el siguiente resultado:
De un total de **30 personas** encuestadas:

- ✓ **20** poseen un conocimiento regular sobre la calidad de software, y **4** admiten tener poco conocimiento del tema; lo que representa un 80% de personas con conocimiento entre medio y bajo y un 20% con conocimientos elevados.

- ✓ **21** poseen un conocimiento regular sobre las pruebas de software, y **5** admite tener poco conocimiento del tema; lo que representa un 87% de personas con conocimiento entre medio y bajo.

- ✓ **20** afirman haber realizado pruebas de software, y **10** plantean que no; lo que representa un 67% de personas que aplican las pruebas.

- ✓ De las **20** personas que plantearon haber realizado pruebas, **11** lo hicieron de Caja Negra, **5** de Caja Blanca, y **4** de documentación, lo que representa un 37% de personas que sólo realizaron pruebas de especificación, 35%, de código y especificación, y sólo un 20% bastante integrales (código, especificación y documentación).

- ✓ **13** consideran como medio el nivel de aplicación de pruebas de software en el proyecto, **15** lo consideran adecuado y **2** no poseen criterio acerca del tema; lo que representa aproximadamente un 43% de personas que cree que existe algún nivel de avance en este sentido, 50% no lo creen así, y un 7% carece de opinión al respecto.

Las causas a que le atribuyen este problema, se distribuyen de la siguiente manera:

- ✓ **4** personas consideran que es debido a la falta de una adecuada estructura organizativa en las empresas para este efecto: 31%.
- ✓ **9** consideran que es escaso el personal capacitado y disponible para ello: 69%.

30 le conceden mucha importancia a la aplicación de las pruebas en su proyecto para un 100%.

Algunos de estos datos estadísticos se reflejan en los siguientes gráficos, para brindar una idea más completa acerca de la situación real:

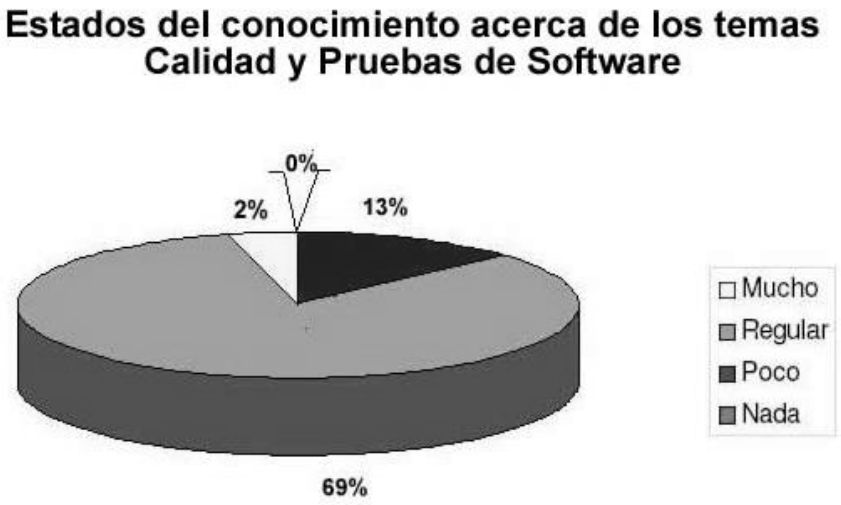


Figura 2. Estado del conocimiento acerca del tema Calidad de Software

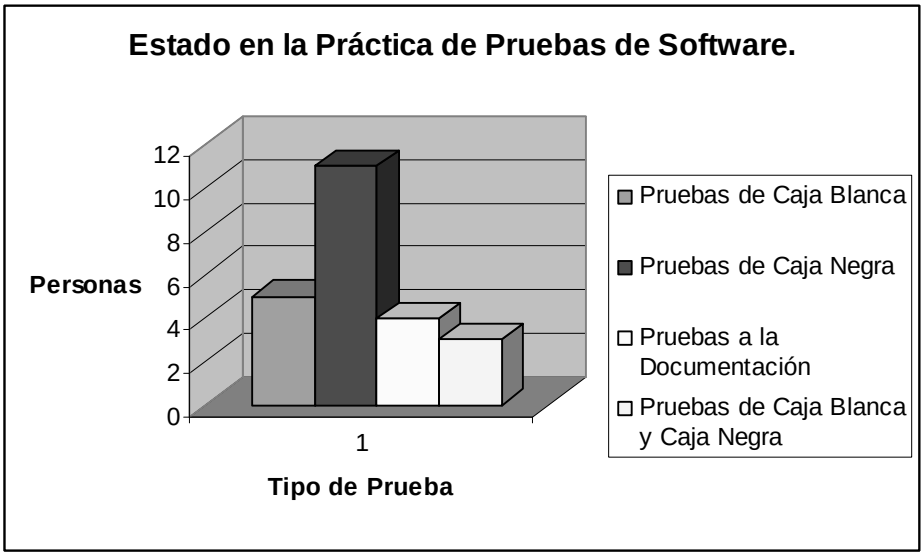


Figura 3. Estado de la Práctica de Pruebas de Software

A partir de los resultados obtenidos se puede concluir que en el grupo de trabajo GPI que implementó el Sistema Cassandra PACSS no existe una vasta cultura sobre el tema de calidad, específicamente de la parte concerniente a las pruebas de software.

Durante el desarrollo del software generalmente no existen personas que cumplan los roles de probadores y por ende en el período en que corresponde probar el sistema no se cumple con lo establecido para ello. A dicho sistema no se le aplican pruebas de caja blanca, y en caso de que se realicen, lo hacen los mismos programadores.

De igual forma sucede con las pruebas de caja negra, aunque estas últimas se realizan con más frecuencia.

Por otra parte la mayoría le confieren mucha importancia a la práctica de las pruebas de software y sólo un por ciento muy bajo ha realizado pruebas de caja blanca alguna vez.

3.3 Diseño de las Pruebas de Software

En un estudio realizado al Grupo de Procesamiento de Imágenes y Señales (GPI), productores de software y autores del sistema Cassandra PACSS del cuál se probará una parte en este trabajo se detectaron una serie de deficiencias en el control de calidad de sus productos.

Por la vital importancia que tiene la revisión exhaustiva del código de un producto de software y porque el sistema seleccionado solo ha tenido revisiones informales del mismo se llevaran a cabo las pruebas de caja blanca usando el Procedimiento propuesto en el capitulo anterior para de esta forma partiendo de que se conoce el funcionamiento del producto se puedan 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 y que por tanto el Procedimiento propuesto cumple sus objetivos. Por razones relacionadas con el tiempo solamente se realizarán las pruebas al Módulo Dmail del sistema antes mencionados.

A este módulo se le hará un minucioso examen de los detalles procedimentales. Se comprobarán los caminos lógicos del mismo proponiendo casos de prueba que examinen que están correctas todas

las condiciones para determinar si el estado real coincide con el esperado o afirmado y de esta manera que cuente entonces con su primera revisión reglamentada y documentada del código y se podrá garantizar su estabilidad y óptimo funcionamiento, además de lograr su comercialización con la mayor calidad requerida.

3.4 Cassandra DMail

Se presenta el sistema Cassandra DICOM Mail, como alternativa a la transmisión de imágenes médicas en el Sistema Nacional de Salud Cubano. El sistema fue desarrollado por el Grupo de Procesamiento Digital de Imágenes y Señales de la Universidad de Ciencias Informáticas, y permite la transmisión de imágenes médicas mediante el uso del protocolo estándar para la comunicación de imágenes digitales en medicina *(DICOM 3.0).*

Dicho sistema promueve un servicio de mensajería conforme a DICOM, que apoyado en una interfaz de correo electrónico que brinda la aplicación cliente, logra abstraer a los usuarios de las complejidades y necesidades de la transmisión conforme al estándar. DICOM posibilita que los archivos médicos puedan viajar de forma segura entre hospitales, centros investigación, etc.

Luego esa información puede ser vista remotamente para que los médicos puedan diagnosticar desde su casa o buscar diferentes opiniones de otros expertos de forma rápida y sencilla. El sistema fue concebido y desarrollado sobre tecnología .NET, por lo que requiere para su funcionamiento del Microsoft .Net Framework SDK v1.1 o superior. Para su implementación se utilizó C# como lenguaje de programación en el entorno Microsoft Visual Studio . 2003, y se trabajó con la librería MyDICOM.NET en cuanto al estándar se refiere.

3.5 Características a probar

Los atributos a medirse varían y su relevancia y el nivel de detalle dependen del propósito de cada aplicación particular y los objetivos de la prueba. Hay algunos atributos que siempre se miden a la hora de desarrollar las pruebas de software, como por ejemplo si la aplicación cumple sus requerimientos y especificaciones, si se ejecuta sin errores, si es fácil de usar, si es aceptable para el

usuario, si la documentación está completa y correcta. También aquellos que determinan si la aplicación es confiable, precisa, completa, rápida, utilizable, flexible y bien documentada.

Además se pueden medir atributos del código fuente para saber cuán bien estructurado y reutilizable es, ya que estos atributos determinarán el futuro de la aplicación. Una aplicación que es difícil de mantener, resultará muy costosa y posiblemente no dure mucho, pues resulte "imposible" de mantener activa.

En el Cassandra DICOM Mail que se prueba en este trabajo se miden principalmente los atributos de confiabilidad, funcionamiento y usabilidad, persiguiendo con la comprobación de estos una aplicación más estable y eficiente.

3.5.1 Requerimientos necesarios para realizar las pruebas

En la realización de las pruebas al Módulo DMail se tuvieron en cuenta algunos requerimientos de funcionalidad del mismo, requerimientos de tiempo, de recursos humanos, recursos tangibles, entre otros.

Para comenzar a probar cada uno de los módulos se esperó a que tuvieran todas las funcionalidades requeridas y que corrieran lo más estable posible, porque no cumple objetivo probar una configuración que aún reporta fallas y por otro lado, si esperamos a que todos los módulos estén perfectos, puede que detectemos fallas graves demasiado tarde.

Para la realización de las pruebas fue necesario constar algunos recursos tangibles dentro de los cuales podemos mencionar los siguientes:
Como mínimo una PC que como requerimiento de **software** debía tener:

- ✓ Windows 95 o superior.
- ✓ Office 97 o superior.
- ✓ Framework 2.0
- ✓ Microsoft Visual Studio 2005
- ✓ Memoria de Video: 128MB

Y como requerimiento de **hardware***:*

- ✓ Microprocesador Pentium o superior
- ✓ Al menos 256 MByte de memoria RAM.
- ✓ Tarjeta de Red.
- ✓ Mínimo de 40 GByte de Disco duro.

La ejecución de las pruebas a los módulos se realizó desde una terminal que se conecta, a través de una red Local a otra Terminal, ambas con el DICOM Mail instalado.

Para diseñar**,** implementar y ejecutar las pruebas fue necesario constar con dos probadores, así como también se necesitó un estimado de tiempo de 2 meses entre ambos probadores, contando 20 días hábiles al mes y 6 horas diarias de tiempo de máquina. Esto, teniendo en cuenta que no se tenía conocimiento del software ni de los temas prácticos de las pruebas de caja blanca.

3.6 Descripción de los Casos de Prueba

Para el diseño de los Casos de Prueba se usa la plantilla propuesta por el procedimiento solo en el primero de ellos y de manera ilustrativa. Ya que en este trabajo de diploma los diseñadores de prueba y probadores son las mismas personas.

Se diseñaron los casos de prueba de manera abreviada, o sea cuando el diseñador de casos de prueba es una persona y el probador es otra los casos deben ser descrito usando la plantilla propuesta para que la comunicación se establezca sin ninguna dificultad . En este caso el probador ya conoce todas las entradas y resultados esperados, porque es el mismo diseñador y no es necesario hacer la descripción extendida de todos los casos.

Vale señalar que en este trabajo no se publica el código fuente del Módulo Dmail del Sistema Cassandra PACS al cual se le aplicaron las pruebas dado que la dirección del proyecto y de la Facultad de las Ciencias Informáticas así lo determinó para la protección y seguridad del software

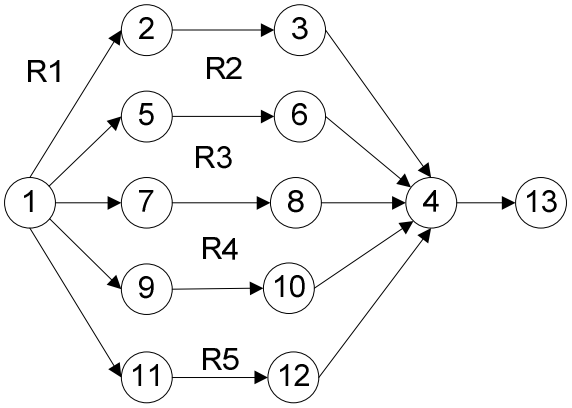
3.6.1 Pruebas de Caja Blanca usando el Procedimiento Propuesto

TITULO : Especificación de Casos de Prueba Nombre del Proyecto: Cassandra PACS. COPIA CONTROLADA N^{O} : 1 ASIGNADA A : Facultad 7 de la Universidad de las Ciencias Informáticas	
REDACTADO POR : Yaimí Márquez Alpízar y Yenni Valdés Hechavarraría	18/04/2007
REVISADO POR : Regla María Silva	25/04/2007
APROBADO POR: Regla María Silva	25/04/2007

1. CB_Mod_DMail_ Welcome Form

Void Load Style

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R5

2da vía: V (G)= 16 - 13 + 2

V (G)= 5

3ra vía: V (G)= 4 + 1

V (G)= 5

Caminos Básicos

Camino 1: 1-2-3-4-13

Camino 2: 1-5-6-4-13

Camino 3: 1-7-8-4-13

Camino 4: 1-9-10-4-13

Camino 5: 1-11-12-4-13

.

2. El usuario selecciona la opción *Inbox* de la Forma de Bienvenida.

2.1. Descripción

El usuario selecciona la opción Inbox de la Forma de Bienvenida y se actualiza la ventana mostrándose la Bandeja de Entrada con todos los mensajes asociados a la misma, se mantiene el resto de las opciones del menú principal.

2.2. Condiciones de ejecución

El DMail debe haberse iniciado correctamente y debe abrirse por defecto la Bandeja de Bienvenida.

2.3. Entrada

Style= “Inbox”

2.4. Resultados Esperados

Se actualiza la información a mostrar. Se busca en el diccionario las palabras en el nuevo lenguaje correspondiente a la cabecera de los mensajes y a la Bandeja de Entrada.

2.5. Evaluación de la Prueba

El sistema responde correctamente a la prueba, los resultados obtenidos coinciden con los esperados y no se encontró ningún error o anomalía durante la realización de la misma

3. El usuario selecciona la opción *Outbox* de la Forma de Bienvenida.

3.1. Descripción

El usuario selecciona la opción Outbox de la Forma de Bienvenida y se actualiza la ventana mostrándose la Bandeja de Salida con todos los mensajes asociados a la misma, se mantiene el resto de las opciones del menú principal.

3.2. Condiciones de ejecución

El DMail debe haberse iniciado correctamente y debe abrirse por defecto la Bandeja de Bienvenida.

3.3. Entrada

Styles= “Outbox”

3.4. Resultados Esperados

Se actualiza la información a mostrar. Se activa y se visualiza la Bandeja de Salida con todos sus elementos.

3.5. Evaluación de la Prueba

El sistema responde correctamente a la prueba, los resultados obtenidos coinciden con los esperados y no se encontró ningún error o anomalía durante la realización de la misma.

4. El usuario selecciona la opción *Elementos Enviados* de la Forma de Bienvenida.

4.1. Descripción

El usuario selecciona la opción ***Send Items*** de la Forma de Bienvenida y se actualiza la ventana mostrándose la Bandeja de Salida con todos los mensajes asociados a la misma, se mantiene el resto de las opciones del menú principal.

4.2. Condiciones de ejecución

El DMail debe haberse iniciado correctamente y debe abrirse por defecto la Bandeja de Bienvenida.

4.3. Entrada

Styles= “Send Items”

4.4. Resultados Esperados

Se actualiza la información a mostrar. Se activa y se visualiza la Bandeja de Elementos Enviados

4.5. Evaluación de la Prueba

El sistema responde correctamente a la prueba, los resultados obtenidos coinciden con los esperados y no se encontró ningún error o anomalía durante la realización de la misma.

5. El usuario selecciona la opción *Elementos Eliminados* de la Forma de Bienvenida.

5.1. Descripción

El usuario selecciona la opción ***Delete Items*** de la Forma de Bienvenida y se actualiza la ventana mostrándose la Bandeja de Salida con todos los mensajes asociados a la misma, se mantiene el resto de las opciones del menú principal.

5.2. Condiciones de ejecución

El DMail debe haberse iniciado correctamente y debe abrirse por defecto la Bandeja de Bienvenida.

5.3. Entrada

Styles= “Delete Items”

5.4. Resultados Esperados

Se actualiza la información a mostrar. Se activa y se visualiza la Bandeja de Elementos Eliminados.

5.5. Evaluación de la Prueba

El sistema responde correctamente a la prueba, los resultados obtenidos coinciden con los esperados y no se encontró ningún error o anomalía durante la realización de la misma.

6. El usuario no selecciona ninguna opción.

6.1. Descripción

La Bandeja de Bienvenida se queda igual porque el usuario no la ha modificado.

6.2. Condiciones de ejecución

El DMail debe haberse iniciado correctamente y debe abrirse por defecto la Bandeja de Bienvenida.

6.3. Entrada

Styles= “Null”

6.4. Resultados Esperados

No se modifica la información a mostrar.5.5. Evaluación de la Prueba

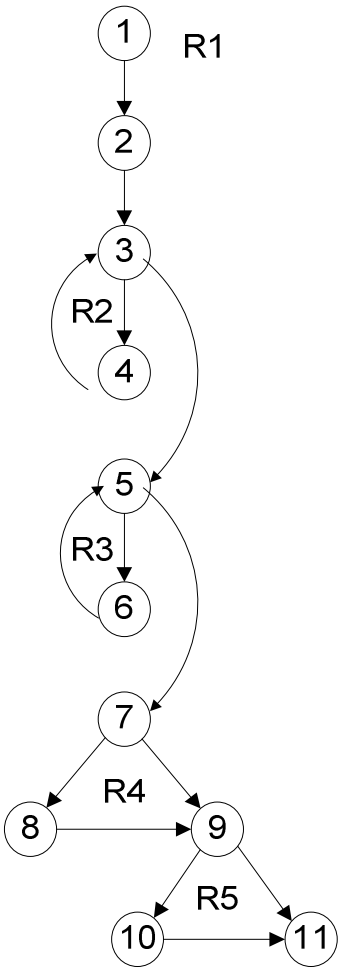
6.5. Evaluación de la Prueba

El sistema responde correctamente a la prueba, los resultados obtenidos coinciden con los esperados y no se encontró ningún error o anomalía durante la realización de la misma.

CB_Mod_DMail_Welcome Form

Void Load Language

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= No.Regiones del Grafo

V (G)= R5

2da vía: V (G)= No.Aristas – No.Nodos + 2

V (G)= 14 – 11 + 2

V (G)= 5

3ra vía: V (G)= No.Nodos Predicados + 1

V (G)= 4 + 1

V (G)= 5

Caminos Básicos

Camino 1: 1-2-3-4-3-5-6-5-7-8-9-10-11

Camino 2: 1-2-3-4-3-5-6-5-7-9-10-11

Camino 3: 1-2-3-4-3-5-6-5-7-9-11

Camino 4: 1-2-3-4-3-5-6-5-7-8-9-11

Camino 5: 1-2-3-5-7-9-11

Casos de Prueba

Caso 1:

Style= “Inbox”

R.E: Se actualiza la información a mostrar. Se busca en el diccionario las palabras en el nuevo lenguaje correspondiente a la cabecera de los mensajes y a la Bandeja de Entrada.

Caso 2:

Rowns.Count! = 0

Style= “Outbox”

R.E: Se actualiza la información a mostrar, se localiza en el diccionario las palabras en el nuevo lenguaje pertenecientes a la Bandeja de Salida que no tiene mensajes.

Caso 3:

Rowns.Count = 0

Style= “Null”

R.E: Se modifica la información a mostrar en la Ventana Principal solamente pues no hay ningún Menú seleccionado.

Caso 4:

Rowns.Count! = 0

Style= “Null”

R.E: Se modifica la información a mostrar, se localiza en el diccionario las palabras en el nuevo lenguaje para la Ventana Principal y los mensajes seleccionados.

Caso 5:

Rowns.Count = 0

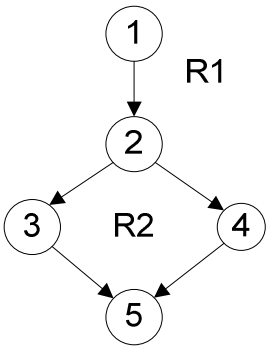
Style= “Null”

R.E: No se modifica la información a mostrar, no se localiza nada en el diccionario.

CB_Mod_DMail_ Welcome Form

Void Reply_Click

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R2

2da vía: V (G)= 5 - 5 + 2

V (G)= 2

3ra vía: V (G)= 1 + 1

V (G)= 2

Caminos Básicos

Camino 1: 1-2-3-5

Camino 2: 1-2-4-5

Casos de Prueba

Caso 1:

Select Rows! =" Null"

R.E: Se actualiza la información a mostrar, se activa y visualiza una nueva ventana en forma de mensaje que corresponde al que estaba seleccionado.

Caso 2:

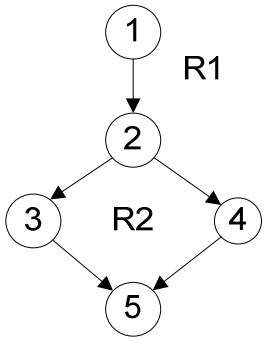
Select Rows=" Null"

R.E: Se muestra un aviso de error por no estar seleccionado ningún mensaje.

CB_Mod_DMail_ Welcome Form

Void Forward_Click

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R2

2da vía: V (G)= 5 - 5 + 2

V (G)= 2

3ra vía: V (G)= 1 + 1

V (G)= 2

Caminos Básicos

Camino 1: 1-2-3-5

Camino 2: 1-2-4-5

Casos de Prueba

Caso 1:

Selection! =" Null"

R.E: Se actualiza la información a mostrar, se activa y visualiza una nueva ventana en forma de mensaje con el mismo contenido que el que estaba seleccionado para reenviarlo.

Caso 2:

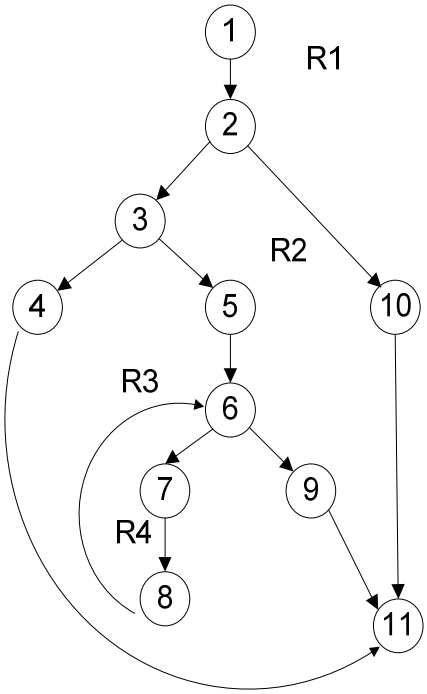
Selection=” Null”

R.E: Se muestra un aviso de error por no estar seleccionado ningún mensaje.

CB_Mod_DMail_ Welcome Form

Void Delete_Click

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R4

2da vía: V (G)= 13 - 11 + 2

V (G)= 4

3ra vía: V (G)= 3 + 1

V (G)= 4

Caminos Básicos

Camino 1: 1-2-3-5-6-7-8-6-7-8-6-9-11

Camino 2: 1-2-10-11

Camino 3: 1-2-3-5-6-7-8-6-9-11

Camino 4: 1-2-3-4-11

Casos de Prueba

Caso 1:

Items! =" Delete"

Selection= "2"

R.E: Se mueven los mensajes seleccionados para los Elementos Eliminados y se restauran los eventos.

Caso 2:

Selection= "Null"

R.E: Se muestra un aviso de error por no estar seleccionado ningún mensaje.

Caso 3:

Items! =" Delete"

Selection= "1"

R.E: Se mueve el mensaje seleccionado para los Elementos Eliminados y se restauran los eventos.

Caso 4:

Items=" Delete"

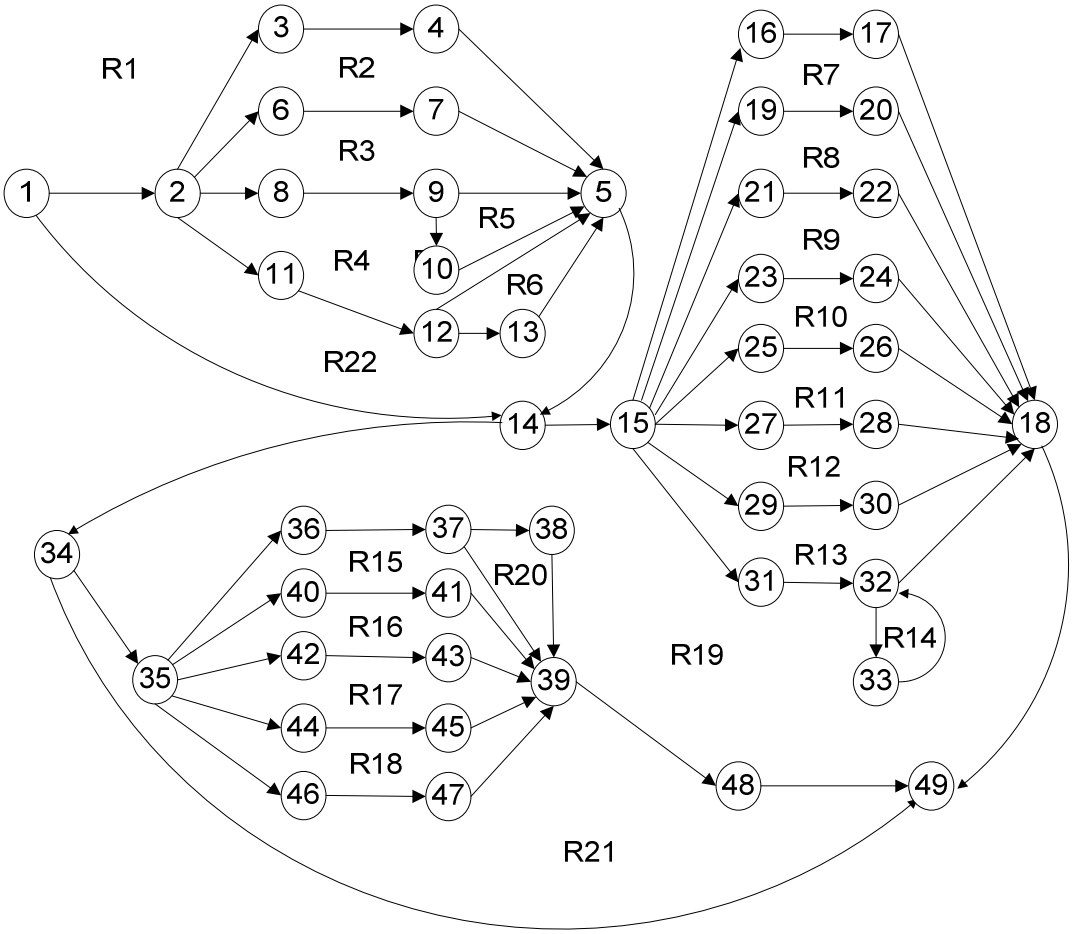
Selection! = "Null"

R.E: Se muestra un aviso de confirmación para eliminar permanentemente los mensajes seleccionados en Elementos Eliminados.

CB_Mod_DMail_ Welcome Form

Void Preview_ Key Dow

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R22

2da vía: V (G)= 69 – 49 + 2

V (G)= 22

3ra vía: V (G)= 21 + 1

V (G)= 22

Caminos Básicos

Camino 1: 1-14-34-49

Camino 2: 1-2-3-4-5-14-34-49

Camino 3: 1-2-6-7-5-14-34-49

Camino 4: 1-2-8-9-5-14-34-49

Camino 5: 1-2-8-9-10-5-14-34-49

Camino 6: 1-2-11-12-13-5-14-34-49

Camino 7: 1-2-11-12-5-14-34-49

Camino 8: 1-14-15-16-17-18-49

Camino 9: 1-14-15-19-20-18-49

Camino 10: 1-14-15-21-22-18-49

Camino 11: 1-14-15-23-24-18-49

Camino 12: 1-14-15-25-26-18-49

Camino 13: 1-14-15-27-28-18-49

Camino 14: 1-14-15-29-30-18-49

Camino 15: 1-14-15-31-32-18-49

Camino 16: 1-14-15-31-32-33-32-18-49

Camino 17: 1-14-34-35-36-37-38-39-48-49

Camino 18: 1-14-34-35-36-37-39-48-49

Camino 19: 1-14-34-35-40-41-39-48-49

Camino 20: 1-14-34-35-42-43-39-48-49

Camino 21: 1-14-34-35-44-45-39-48-49

Camino 22: 1-14-34-35-46-47-39-48-49

Casos de Prueba

Caso 1:

R.E: No se modifica la información a mostrar porque no hay teclas de control presionadas.

Caso 2:

Keys = “F5”

R.E: Se actualiza la ventana que está activada en ese momento.

Caso 3:

Keys = “Escape”

R.E: Se cierra la ventana que está activada en ese momento.

Caso 4:

Keys = “Enter”

Selected Row = 0

R.E: No se modifica la información a mostrar pues no existe mensaje seleccionado.

Caso 5

Keys = “Enter”

Selected Row! = 0

R.E: Se modifica la información a mostrar. Se abre una nueva ventana con el mensaje seleccionado.

Caso 6:

Keys = “Delete”

R.E: Se mueve el mensaje seleccionado para los Elementos Eliminados y se restauran los eventos.

Caso 7:

Keys = “Shift + Delete”

R.E: Se modifica la información a mostrar. Se muestra un aviso de confirmación para eliminar permanentemente los mensajes seleccionados.

Caso 8:

Keys = “Control + N”

R.E: Se actualiza la información a mostrar y se abre una nueva ventana en forma de mensaje.

Caso 9:

Keys = “Control + R”

R.E: Se actualiza la información a mostrar y se abre una nueva ventana en forma de mensaje que responde al que estaba seleccionado.

Caso 10:

Keys = “Control + F”

R.E: Se actualiza la información a mostrar y se abre una nueva ventana en forma de mensaje con el contenido del mensaje seleccionado.

Caso 11:

Keys = “Control + O”

R.E: Se mueve el mensaje seleccionado para los Elementos Eliminados y se restauran los eventos.

Caso 12:

Keys = “Control + C”

R.E: Se modifica la información a mostrar y se abre la ventana de Contacto.

Caso 13:

Keys = “Control + S”

R.E: Se salva el mensaje que estaba abierto.

Caso 14:

Keys = “Control + P”

R.E: Se modifica la información a mostrar y se abre la ventana de cambiar contraseña.

Caso 15:

Keys = “Control + A”

R.E: No se modifica la información a mostrar.

Caso 16:

Keys = “Control + A”

R.E: Se modifica la información a mostrar y se abre una nueva ventana en forma de mensaje.

Caso 17:

Keys = “Shift + Delete”

R.E: Se modifica la información a mostrar y se muestra un aviso de confirmación para eliminar permanentemente los mensajes seleccionados

Caso 18:

Keys = “Shift + Delete”

R.E: No se modifica la información a mostrar por no existir mensaje seleccionado.

Caso 19:

Keys = “Shift + I”

R.E: Se actualiza la información a mostrar, se activa y visualiza la Bandeja de Entrada con todos sus elementos.

Caso 20:

Keys = “Shift + O”

R.E: Se actualiza la información a mostrar, se activa y visualiza la Bandeja de Salida con todos sus elementos.

Caso 21:

Keys = “Shift + S”

R.E: Se actualiza la información a mostrar, se activa y visualiza la Bandeja de Elementos Enviados.

Caso 22:

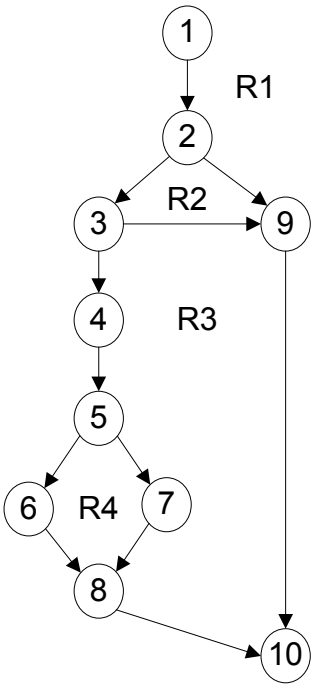
Keys = “Shift + D”

R.E: Se actualiza la información a mostrar, se activa y visualiza la Bandeja de Elementos Eliminados.

CB_Mod_DMail_ Welcome Form

Void DICOM Preview_Click

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R4

2da vía: V (G)= 12 – 10 + 2

V (G)= 4

3ra vía: V (G)= 3 + 1

V (G)= 4

Caminos Básicos

Camino 1: 1-2-8-9-10

Camino 2: 1-2-3-4-5-7-10

Camino 3: 1-2-3-4-5-6-8-10

Camino 4: 1-2-3-8-9-10

Casos de Prueba

Caso 1:

Preview.Image = “Null”

R.E: Se cambia la información a mostrar, se visualiza un mensaje de error por no existir imagen previa para mostrar.

Caso 2:

Preview.Image! = “Null”

Selected = 1

Path = “Null”

R.E: Se modifica la información a mostrar, se enseña un mensaje de error por no encontrarse el camino a la imagen anterior.

Caso 3:

Preview.Image! = “Null”

Selected = 1

Path! = “Null”

R.E: Se modifica la información a mostrar, se localiza la imagen previa, se muestra en el DataGrid y se restauran los eventos.

Caso 4:

Preview.Image! = “Null”

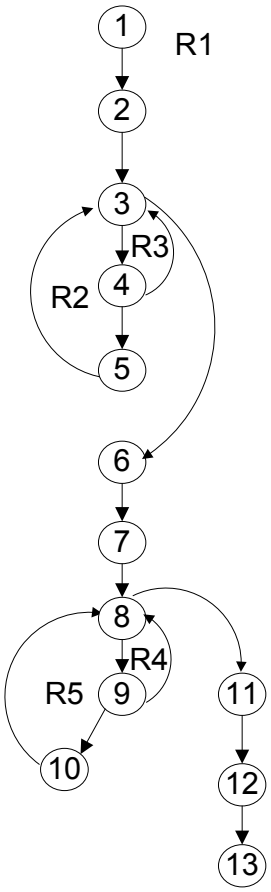
Selected = 0

R.E: Se modifica la información a mostrar, se visualiza un mensaje de error por no estar seleccionado ningún mensaje.

CB_Mod_DMail_ Welcome Form

Void Cassandra Installer Path

Grafo de Flujo



Complejidad Ciclomática

1ra Vía: V (G) = R5

2da Vía: V (G) = 16 – 13 + 2

V (G) = 5

3ra Vía: V (G) = 4 + 1

V (G) = 5

Caminos Básicos

Camino 1: 1-2-3-4-3-6-7-8-9-8-11-12-13

Camino 2: 1-2-3-4-5-3-6-8-9-10-11-12-13

Camino 3: 1-2-3-4-3-6-7-8-9-10-11-12-13

Camino 4: 1-2-3-4-5-3-6-8-9-8-11-12-13

Camino 5: 1-2-3-6-7-8-11-12-13

Casos de Prueba

Caso 1:

Path_Installer\\Assemblies =”False”

Path_Software\\Microsoft\\Installer\\Assemblies =”True”

R.E: Se modifica la información a mostrar y se muestra el camino de acceso al Cassandra Viewer reemplazándose los “l “por” \.”

Caso 2:

Path_Installer\\Assemblies =”True”

Path_Software\\Microsoft\\Installer\\Assemblies =” False”

R.E: Se modifica la información a mostrar y se muestra el camino de acceso al Cassandra Viewer reemplazándose los “l “por” \.”

Caso 3:

Path_Installer\\Assemblies =”False”

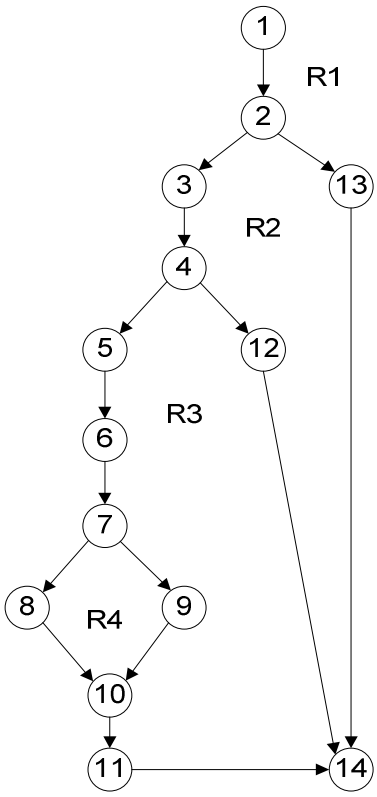
Path_Software\\Microsoft\\Installer\\Assemblies =” False”

R.E: No se encuentra el camino para abrir el Cassandra Viewer pues no se encuentra instalado al sistema.

CB_Mod_DMail_ Welcome Form

Void Editor_CurrentCellChanged

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R4

2da vía: V (G)= 16 – 14 + 2

V (G)= 4

3ra vía: V (G)= 3 + 1

V (G)= 4

Caminos Básicos

Camino 1: 1-2-3-4-5-6-7-8-10-11-14

Camino 2: 1-2-13-14

Camino 3: 1-2-3-4-5-6-7-9-10-11-14

Camino 4: 1-2-3-4-12-14

Casos de Prueba

Caso 1:

Selected Rows Count = 1

Node! = “Null”

Node.Attachment! =”Null”

R.E: Se modifica la información a mostrar. Se carga del XML el contenido de nuevo mensaje a mostrar, se localiza en el diccionario las palabras en el lenguaje correspondiente. Se guarda el DICOM del mensaje que se va a quitar en Previous y se cambia de mensaje mostrando el nuevo con todo su contenido.

Caso 2:

Selected Rows Count = 0

R.E: No se modifica la información a mostrar.

Caso 3:

Selected Rows Count = 1

Node! = “Null”

Node. Attachment =”Null”

R.E: Se modifica la información a mostrar, se carga del XML el contenido de nuevo mensaje, se localiza en el diccionario las palabras en el lenguaje correspondiente y se cambia para el otro mensaje mostrando el mismo con todo su contenido.

Caso 4:

Selected Rows Count = 1

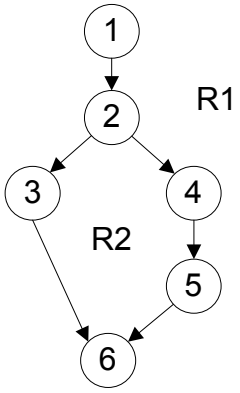
Node = “Null”

R.E: No se modifica la información a mostrar.

CB_Mod_DMail_ Setting Form

Void ComboBoxTransmission_SelectedIndexChanged

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R2

2da vía: V (G)= 6 – 6 + 2

V (G)= 2

3ra vía: V (G)= 1 + 1

V (G)= 2

Caminos Básicos

Camino 1: 1-2-3-6

Camino 2: 1-2-4-5-6

Casos de Prueba

Caso 1:

SelectedItem! =”SelectedDICOM”

R.E: Se ponen las letras del botón Buscar Archivo de color negro y se habilita el ComboBox de búsqueda.

Caso 2

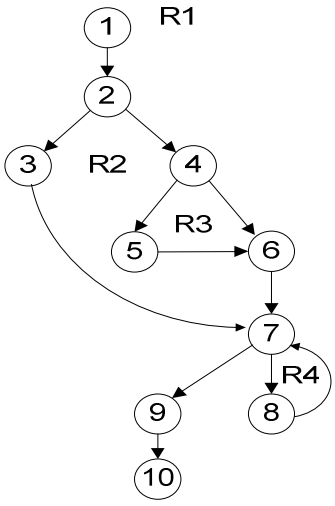
SelectedItem =”SelectedDICOM”

R.E: Se ponen las letras del botón Buscar Archivo de color gris claro y se deshabilita el ComboBox de búsqueda.

CB_Mod_DMail_ Setting Form

Void Update Component

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R4

2da vía: V (G)= 12 – 10 + 2

V (G)= 4

3ra vía: V (G)= 3 + 1

V (G)= 4

Caminos Básicos

Camino 1: 1-2-3-7-8-7-9-10

Camino 2: 1-2-4-5-6-7-8-7-9-10

Camino 3: 1-2-4-6-7-8-7-9-10

Camino 4: 1-2-3-7-9-10

Casos de Prueba

Caso 1:

Logged Account_Directory = “None”

PathExist = “False”

R.E: Se localiza el camino hacia el fichero DICOM en la carpeta del Directorio DICOM. Se salva el DICOM y se actualizan los componentes

Caso 2:

Logged Account_Directory! = “None”

R.E: Se añade el camino hacia la carpeta del Directorio DICOM pues el mismo no está definido aún y se actualizan los componentes.

Caso 3:

Logged Account_Directory = “None”

PathExist = “True”

R.E: Se actualizan los componentes.

Caso 4:

Logged Account_Directory = “None”

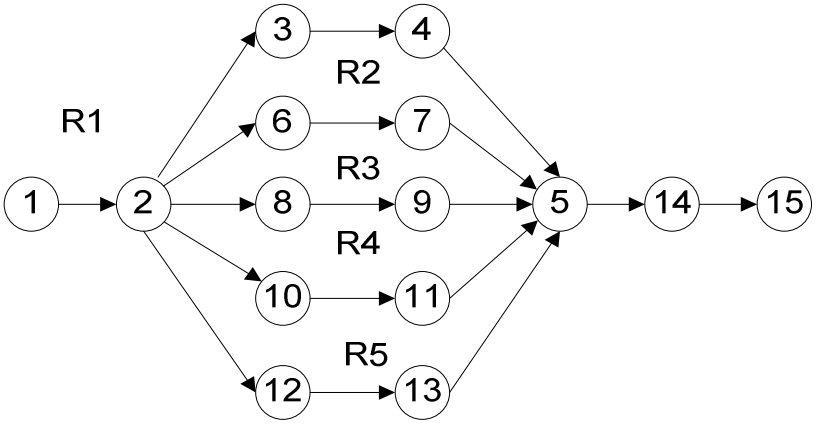
PathExist = “False”

R.E: Se localiza el camino hacia el fichero DICOM en la carpeta del Directorio DICOM. Se salva el DICOM y se actualizan los componentes exceptuando los botones.

CB_Mod_DMail_ Message Form

Void Load Style

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R5

2da vía: V (G)= 18 – 15 + 2

V (G)= 5

3ra vía: V (G)= 4 + 1

V (G)= 5

Caminos Básicos

Camino 1: 1-2-3-4-5-14-15

Camino 2: 1-2-6-7-5-14-15

Camino 3: 1-2-8-9-5-14-15

Camino 4: 1-2-10-11-5-14-15

Camino 5: 1-2-12-13-5-14-15

Casos de Prueba

Caso 1:

Styles = “New Message”

R.E: Se actualiza la información a mostrar. Se activa y visualiza una nueva ventana en forma de mensaje.

Caso 2:

Styles = “Reply”

R.E: Se actualiza la información a mostrar, se activa y visualiza una nueva ventana en forma de mensaje para responder al que estaba seleccionado y se mantiene el texto del mismo.

Caso 3:

Styles = “Forward Styles”

R.E: Se actualiza la información a mostrar, se activa y visualiza una nueva ventana en forma de mensaje con todo el contenido del que se va a reenviar.

Caso 4:

Styles = “Read”

R.E: Se modifica la información a mostrar, se marca el mensaje seleccionado como que ya está leído.

Caso 5:

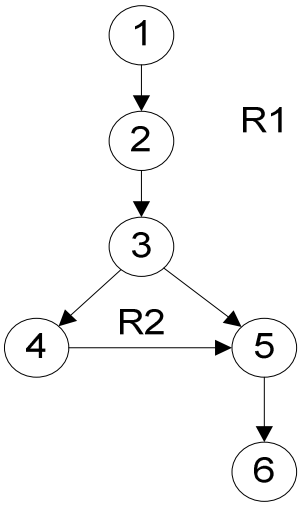
Styles = “Null”

R.E: Se lanza una excepción por no haberse seleccionado un estilo correcto.

CB_Mod_DMail_ Message Form

Void Attach_Click

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R2

2da vía: V (G)= 6 – 6 + 2

V (G)= 2

3ra vía: V (G)= 1 + 1

V (G)= 2

Caminos Básicos

Camino 1: 1-2-3-4-5-6

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

Casos de Prueba

Caso 1:

FileName! = “Null”

R.E: Se actualiza la información a mostrar, se abre una nueva ventana ofreciéndole al usuario las opciones de adjuntar y se muestran excepciones con los errores ocurridos durante la operación.

Caso 2:

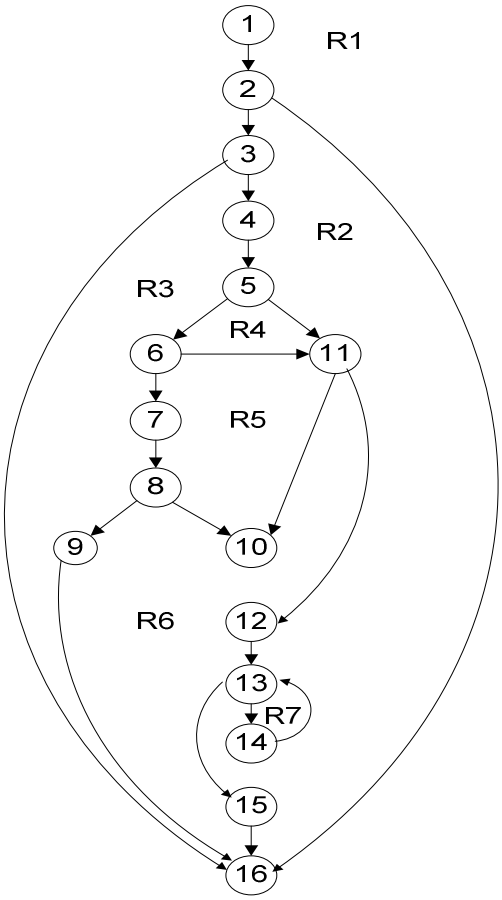
FileName = “Null”

R.E: Se muestra excepciones con los errores encontrados.

CB_Mod_DMail_ Message Form

Void Send Mail

Grafo de Flujo



Complejidad Ciclomática

1ra Vía: V (G) = R7

2da Vía: V (G) = 21 – 16 + 2

V (G) = 7

3ra Vía: V (G) = 6 + 1

V (G) = 7

Caminos Básicos

Camino 1: 1-2-16

Camino 2: 1-2-3-16

Camino 3: 1-2-3-4-5-6-7-8-9-16

Camino 4: 1-2-3-4-5-11-12-13-14-13-15-16

Camino 5: 1-2-3-4-5-6-11-12-13-14-13-15-16

Camino 6: 1-2-3-4-5-6-7-8-10-11-12-13-14-13-15-16

Camino 7: 1-2-3-4-5-6-7-8-10-11-12-13-15-16

Casos de Prueba

Caso 1:

Anonymized!= “Acepted”

R.E: No se envía el mensaje porque no se aceptó el proceso de anonimización.

Caso 2:

Anonymized = “Acepted”

Transfer =”Null”

R.E: No se envía el mensaje porque no se encontró el transfer o se encontró vacío.

Caso 3:

Anonymized = “Acepted”

Transfer! =”Null”

Logged Account Transmisión = “Selected DICOM”

R.E: Se envía el mensaje con el DICOM adjunto.

Caso 4:

Anonymized = “Acepted”

Transfer! =”Null”

SearchScope! = “Custom Folder”

R.E: Se envía el mensaje con las imágenes DICOM que están en el Directorio DICOM.

Caso 5:

Anonymized = “Acepted”

Transfer! =”Null”

SearchScope = “Custom Folder”

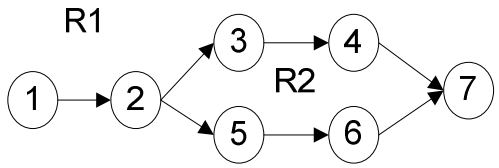
Dialog Result! = “OK”

R.E: No se envía el mensaje porque no se envía el camino hasta la carpeta donde están situados los DICOM que se desean adjuntar.

CB_Mod_DMail_Check Names Form

Void Editor_KeyDown

Grafo de Flujo



Complejidad Ciclomática

1ra Vía: V (G) = R2

2da Vía: V (G) = 7 – 7 + 2

V (G) = 2

3ra Vía: V (G) = 1 + 1

V (G) = 2

Caminos Básicos

Camino 1: 1-2-3-4-7

Camino 2: 1-2-5-6-7

Casos de Prueba

Caso 1:

Keys =”Enter”

R.E: Se acepta lo que está seleccionado en la ventana y se cierra la misma guardando las modificaciones.

Caso 2:

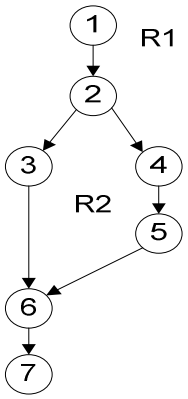
Keys =”Escape”

R.E: Se cancela lo que estaba seleccionado y no se guardan las modificaciones.

CB_Mod_DMail_Check Names Form

Void Editor_DoubleClick

Grafo de Flujo



Complejidad Ciclomática

1ra Vía: V (G) = R2

2da Vía: V (G) = 7 – 7 + 2

V (G) = 2

3ra Vía: V (G) = 1 + 1

V (G) = 2

Caminos Básicos

Camino 1: 1-2-3-6-7

Camino 2: 1-2-4-5-6-7

Casos de Prueba

Caso 1:

Index =”-1”

R.E: Se muestra un mensaje de error por seleccionarse un usuario no válido.

Caso 2:

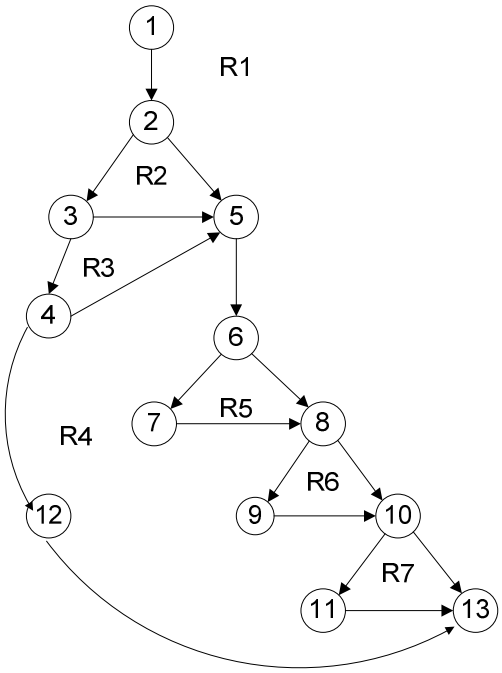
Index =”0”

R.E: Se modifica el mensaje a mostrar y se muestra la información del nuevo usuario seleccionado.

CB_Mod_DMail_ Sending Form

Region Loading Configuration Options

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R7

2da vía: V (G)= 18 – 13 + 2

V (G)= 7

3ra vía: V (G)= 6 + 1

V (G)= 7

Caminos Básicos

Camino 1: 1-2-5-6-7-8-9-10-11-13

Camino 2: 1-2-3-4-12-13

Camino 3: 1-2-3-5-6-8-9-10-11-13

Camino 4: 1-2-5-6-7-8-10-11-13

Camino 5: 1-2-5-6-7-8-9-10-13

Camino 6: 1-2-5-6-7-8-10-13

Camino 7: 1-2-3-4-5-6-8-10-11-13

Casos de Prueba

Caso 1:

Logged Account= “Patient”

Logged Account= “Institution”

Logged Account= “Study”

R.E: Se modifica el mensaje a mostrar y se anonimizan los datos del paciente, de la institución y del estudio.

Caso 2:

Logged Account= “Null”

R.E: Se modifica el mensaje y no se anonimiza ningún parámetro.

Caso 3:

Logged Account= “Institution”

Logged Account= “Study”

R.E: Se modifica el mensaje y se anonimizan los datos de la institución y del estudio.

Caso 4:

Logged Account= “Patient”

Logged Account= “Study”

R.E: Se modifica el mensaje y se anonimizan los datos del paciente y del estudio.

Caso 5:

Logged Account= “Patient”

Logged Account= “Institution”

R.E: Se modifica el mensaje y se anonimizan los datos del paciente y de la institución.

Caso 6:

Logged Account= “Patient”

R.E: Se modifica el mensaje a mostrar y se anonimizan los datos del paciente.

Caso 7:

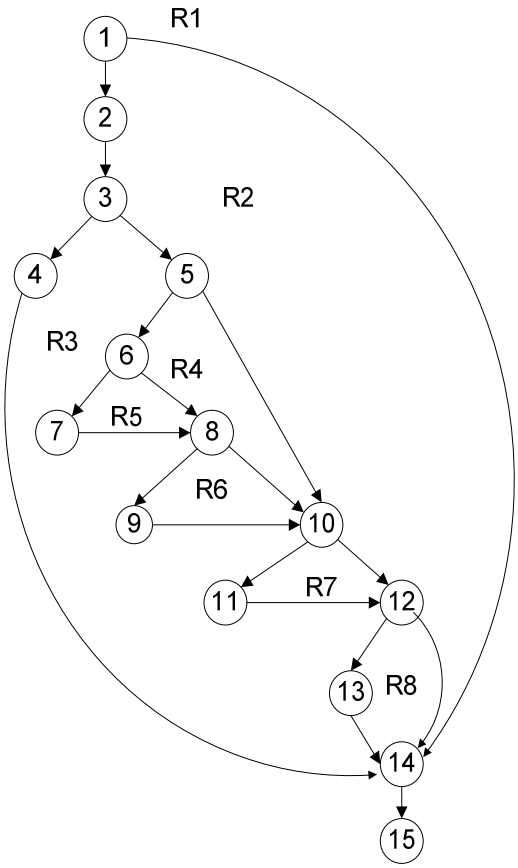
Logged Account= “Study”

R.E: Se modifica el mensaje a mostrar y se anonimizan los datos del estudio.

CB_Mod_DMail_ Sending Form

Region Loading Files

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R8

2da vía: V (G)= 21 - 15 + 2

V (G)= 8

3ra vía: V (G)= 7 + 1

V (G)= 8

Caminos Básicos

Camino 1: 1-14-15

Camino 2: 1-2-3-4-14-15

Camino 3: 1-2-3-5-6-7-8-10-11-12-14-15

Camino 4: 1-2-3-5-6-7-8-10-12-13-14-15

Camino 5: 1-2-3-5-6-8-9-10-11-12-14-15

Camino 6: 1-2-3-5-6-8-9-10-12-13-14-15

Camino 7: 1-2-3-5-10-11-12-14

Camino 8: 1-2-3-5-10-12-13-14

Camino 9: 1-2-3-5-6-8-10-12-14-15

Casos de Prueba

Caso 1:

Attachment= “Null”

R.E: No se modifica el mensaje a mostrar, ni se adjunta nada.

Caso 2:

Attachment! =” Null”

Transmission= “DICOM”

R.E: Se modifica el mensaje a mostrar, se adjunta la imagen DICOM seleccionada y se adiciona el tiempo que demoró el proceso de adjuntar.

Caso 3:

Attachment! =” Null”

SearchScope= “Attachment Folder”

Transmission= “Complete Serie”

R.E: Se modifica el mensaje a mostrar, se localiza la carpeta adjuntada donde se encuentra la Serie Completa a transmitir y se adiciona el tiempo que demoró el proceso de adjuntar.

Caso 4:

Attachment! =" Null"

SearchScope= "Attachment Folder"

Transmission= "Medical Study"

R.E: Se modifica el mensaje a mostrar, se localiza la carpeta adjuntada donde se encuentra el Estudio Médico a transmitir y se adiciona el tiempo que demoró el proceso de adjuntar.

Caso 5:

Attachment! =" Null"

SearchScope= "DICOM Directory"

Transmission= "Complete Serie"

R.E: Se modifica el mensaje a mostrar, se localiza la Serie Completa a transmitir en el Directorio definido para las imágenes DICOM y se adiciona el tiempo que demoró el proceso de adjuntar.

Caso 6:

Attachment! =" Null"

SearchScope= "DICOM Directory"

Transmission= "Medical Study"

R.E: Se modifica el mensaje a mostrar, se localiza el Estudio Médico a transmitir en el Directorio definido para las imágenes DICOM y se adiciona el tiempo que demoró el proceso de adjuntar.

Caso 7:

Attachment! =" Null"

SearchScope= "D\ Imágenes"

Transmission= “Complete Serie”

R.E: Se modifica el mensaje a mostrar, se localiza la carpeta personalizada donde se encuentra la Serie Completa a transmitir y se adiciona el tiempo que demoró el proceso de adjuntar.

Caso 8:

Attachment! =” Null”

SearchScope= “C\ Desktop\ Prueba_Tesis”

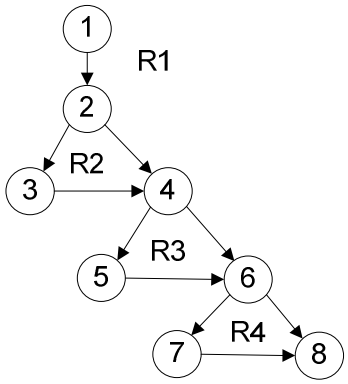
Transmission= “Medical Study”

R.E: Se modifica el mensaje a mostrar, se localiza la carpeta personalizada donde se encuentra el Estudio Médico a transmitir y se adiciona el tiempo que demoró el proceso de adjuntar.

CB_Mod_DMail_ Sending Form

Void Anonymized Institution

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R4

2da vía: V (G)= 10 - 8 + 2

V (G)= 4

3ra vía: V (G)= 3 + 1

V (G)= 4

Caminos Básicos

Camino 1: 1-2-4-6-8

Camino 2: 1-2-3-4-6-8

Camino 3: 1-2-3-4-5-6-8

Camino 4: 1-2-3-4-5-6-7-8

Casos de Prueba

Caso 1:

Anonymized Institution= “Null”

R.E: No se anonimiza ningún dato de la Institución, no se modifica el XML a enviar.

Caso 2:

Anonymized Institution = “Name”

R.E: Se anonimiza el Nombre de la Institución y se elimina el mismo del DICOM a enviar.

Caso 3:

Anonymized Institution = “Name”,”Referring_Physicians”

R.E: Se anonimiza el Nombre y la Referencia Física de la Institución y se eliminan estos dos parámetros del DICOM a transmitir.

Caso 4:

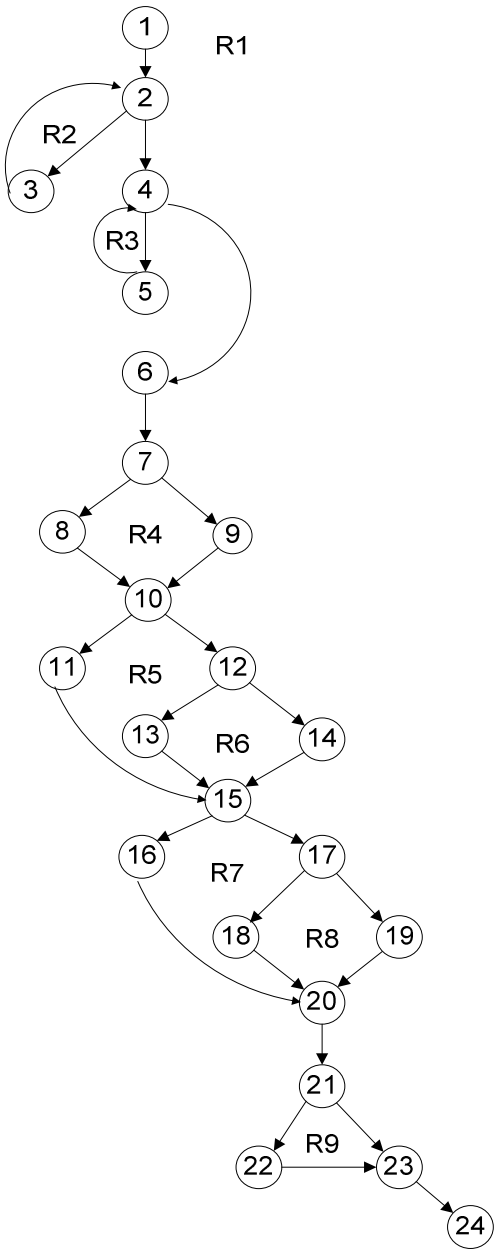
Anonymized Institution = “Name”,”Referring_Physicians”,”Admitting_Diagnoses_ Description”

R.E: Se anonimiza el Nombre, la Referencia Física y la Descripción del Diagnóstico de Admisión de la Institución y se eliminan estos datos de la información del DICOM a transmitir.

CB_Mod_DMail_ Sending Form

Region Sending Files

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R9

2da vía: V (G)= 31- 24 + 2

V (G)= 9

3ra vía: V (G)= 8 + 1

V (G)= 9

Caminos Básicos

Camino 1: 1-2-3-2-4-5-4-6-7-8-10-12-13-15-17-18-20-21-23-24

Camino 2: 1-2-3-2-4-6-7-8-10-12-13-15-17-18-20-21-23-24

Camino 3: 1-2-3-2-4-5-4-6-7-9-10-12-13-15-17-18-20-21-23-24

Camino 4: 1-2-3-2-4-5-4-6-7-8-10-11-15-17-18-20-21-23-24

Camino 5: 1-2-3-2-4-5-4-6-7-8-10-12-14-15-17-18-20-21-23-24

Camino 6: 1-2-3-2-4-5-4-6-7-9-10-12-14-16-20-21-23-24

Camino 7: 1-2-3-2-4-5-4-6-7-8-10-12-13-15-17-19-20-21-23-24

Camino 8: 1-2-3-2-4-5-4-6-7-8-10-12-13-15-17-18-20-21-22-23-24

Camino 9: 1-2-3-2-4-5-4-6-7-9-10-12-14-15-17-19-20-21-23-24

Casos de Prueba

Caso 1:

Transmission= “DICOM”

R.E: Se modifica el mensaje a mostrar, se transfiere el DICOM exitosamente y con todos sus atributos.

Caso 2:

Transmission = “Complete Serie”

R.E: Se modifica el mensaje a mostrar, se transfiere la Serie de DICOM completa con un segundo atributo cada uno.

Caso 3:

Transmission = “DICOM”

Modality= “UnKnown”

R.E: Se modifica el mensaje a mostrar, se inicia la transferencia del DICOM y se especifica que su modalidad es desconocida.

Caso 4:

Transmission = “Medical Study”

R.E: Se modifica el mensaje a mostrar, se transfiere el Estudio Médico con todos sus DICOM y los atributos de cada uno de ellos.

Caso 5:

Transmission = “Complete Serie”

Type= “UnKnown”

R.E: Se modifica el mensaje a mostrar, se transfiere la Serie Completa de DICOM completa y sus atributos especificando que no se conoce el tipo.

Caso 6:

Transmission = “DICOM”

Modality= “UnKnown”

Type= “UnKnown”

R.E: Se modifica el mensaje a mostrar, se transfiere el DICOM sin modalidad ni tipo.

Caso 7:

Transmission = “DICOM”

InputName= “UnKnown”

R.E: Se modifica el mensaje a mostrar, se transfiere el DICOM con un nombre de entrada desconocida.

Caso 8:

Transmission = “DICOM”

R.E: Se modifica el mensaje a mostrar, no se transfiere el DICOM por error en la transmisión.

Caso 9:

Transmission = “Complete Serie”

Modality= “UnKnown”

Type= “UnKnown”

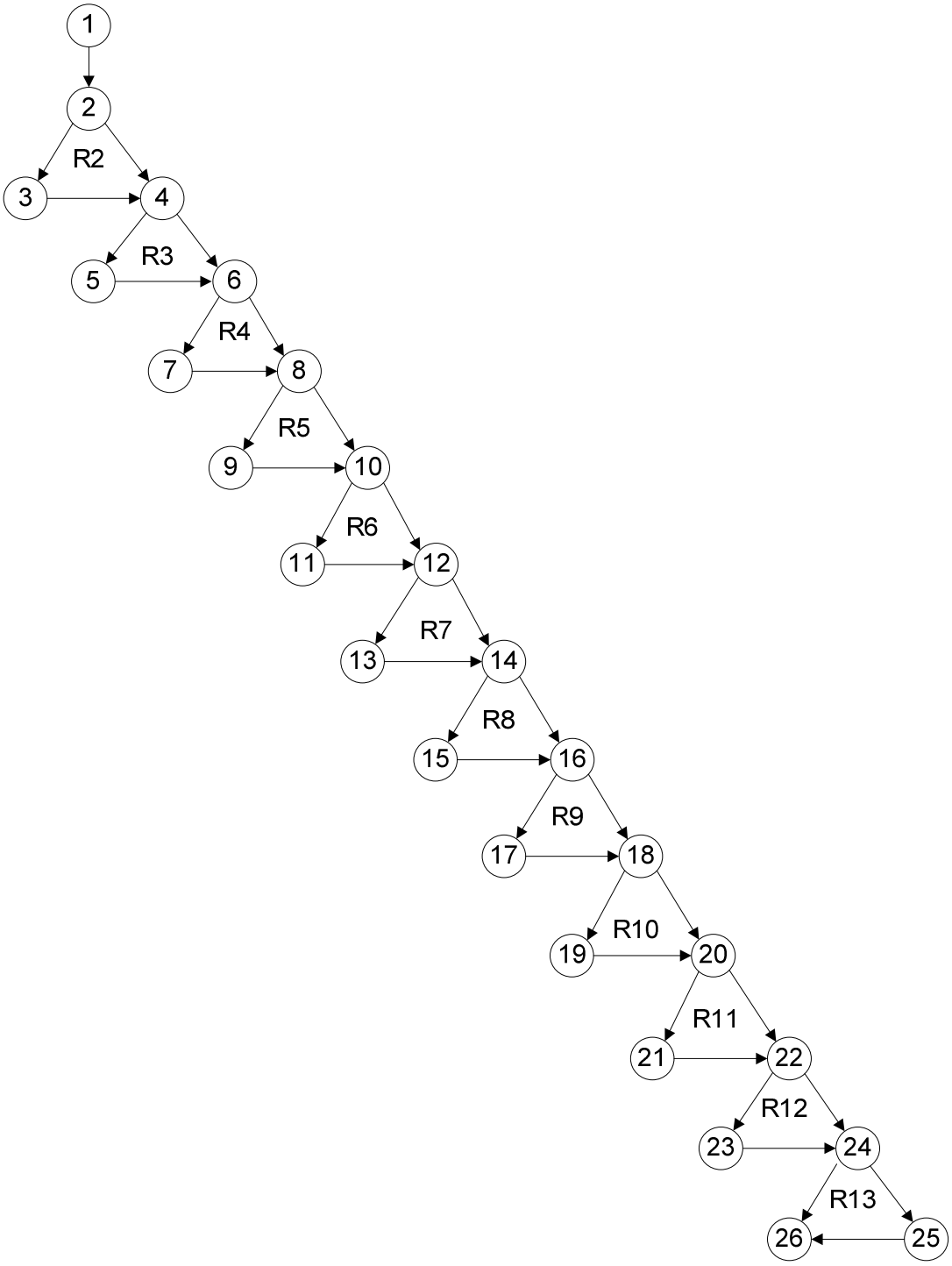
Size= “UnKnown”

R.E: Se modifica el mensaje a mostrar, se transfiere la Serie Completa con una Modalidad, Tipo y Nombre de entrada de los DICOM desconocidos.

CB_Mod_DMail_ Sending Form

Void Anonymized Patient

Grafo de Flujo


1
2
R2
3
4
R3
5
6
R4
7
8
R5
9
10
R6
11
12
R7
13
14
R8
15
16
R9
17
18
R10
19
20
R11
21
22
R12
23
24
R13
26
25

Complejidad Ciclomática

1ra vía: V (G)= R13

2da vía: V (G)= 37- 26 + 2

V (G)= 13

3ra vía: V (G)= 12 + 1

V (G)= 13

Caminos Básicos

Caminos 1: 1-2-4-6-8-10-12-14-16-18-20-22-24-26

Caminos 2: 1-2-3-4-6-8-10-12-14-16-18-20-22-24-26

Caminos 3: 1-2-3-4-5-6-8-10-12-14-16-18-20-22-24-26

Caminos 4: 1-2-3-4-5-6-7-8-10-12-14-16-18-20-22-24-26

Caminos 5: 1-2-3-4-5-6-7-8-9-10-12-14-16-18-20-22-24-26

Caminos 6: 1-2-3-4-5-6-7-8-9-10-11-12-14-16-18-20-22-24-26

Caminos 7: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-16-18-20-22-24-26

Caminos 8: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-18-20-22-24-26

Caminos 9: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-20-22-24-26

Caminos 10: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-22-24-26

Caminos 11: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-24-26

Caminos 12: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-26

Caminos 13: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26

Casos de Prueba

Caso 1:

Anonymized Patient= “Null”

R.E: No se anonimiza ningún dato del Paciente, no se modifica el DICOM a enviar.

Caso 2:

Anonymized Patient= “Name”

R.E: Se anonimiza el Nombre del Paciente y se elimina del DICOM que se va a enviar.

Caso 3:

Anonymized Patient= “Name”, “Id”

R.E: Se anonimiza el Nombre y el Id del Paciente y se eliminan del DICOM que se va a enviar.

Caso 4:

Anonymized Patient= “Name”,”Id”,”Birth_date”

R.E: Se anonimiza el Nombre, el Id y la fecha de nacimiento del Paciente y se eliminan estos datos del DICOM a enviar.

Caso 5:

Anonymized Patient= “Name”,”Id”,”Birth_date”,”Birth_time”

R.E: Se anonimiza el Nombre, el Id, la fecha y la hora de nacimiento del paciente. Se eliminan estos datos del DICOM a enviar.

Caso 6:

Anonymized Patient= “Name”,”Id”,”Birth_date”,”Birth_time”,”Sex”

R.E: Se anonimiza el Nombre, el Id, la fecha y hora de nacimiento y el sexo del paciente. Se eliminan estos datos del DICOM a enviar.

Caso 7:

Anonymized Patient= “Name”,”Id”,”Birth_date”,”Birth_time”,”Sex”,”Other_Names”

R.E: Se anonimiza el Nombre, el Id, la fecha y hora de nacimiento, el sexo y otros nombres del paciente. Se eliminan estos datos del DICOM a enviar.

Caso 8:

Anonymized Patient= “Name”,”Id”,”Birth_date”,”Birth_time”,”Sex”,”Other_Names”,

”Mothers_birth_name”

R.E: Se anonimiza el Nombre, el Id, la fecha y hora de nacimiento, el sexo, otros nombres y el nombre de la madre del paciente. Se eliminan estos datos del DICOM a enviar.

Caso 9:

Anonymized Patient= “Name”,”Id”,”Birth_date”,”Birth_time”,”Sex”,”Other_Names”,

”Mothers_birth_name”, “Military_Rank”

R.E: Se anonimiza el Nombre, el Id, la fecha y hora de nacimiento, el sexo, otros nombres, el nombre de la madre y el Rango Militar del paciente. Se eliminan estos datos del DICOM a enviar.

Caso 10:

Anonymized Patient= “Name”,”Id”,”Birth_date”,”Birth_time”,”Sex”,”Other_Names”,

”Mothers_birth_name”, “Military_Rank”, “Ethnic_group”

R.E: Se anonimiza el Nombre, el Id, la fecha y hora de nacimiento, el sexo, otros nombres, el nombre de la madre, el Rango Militar y el Grupo étnico del paciente. Se eliminan estos datos del DICOM a enviar.

Caso 11:

Anonymized Patient= “Name”,”Id”,”Birth_date”,”Birth_time”,”Sex”,”Other_Names”,

”Mothers_birth_name”, “Military_Rank”, “Ethnic_group”, “Comments”

R.E: Se anonimiza el Nombre, el Id, la fecha y hora de nacimiento, el sexo, otros nombres, el nombre de la madre, el Rango Militar, el Grupo étnico y comentarios del paciente. Se eliminan estos datos del DICOM a enviar.

Caso 12:

Anonymized Patient= “Name”,”Id”,”Birth_date”,”Birth_time”,”Sex”,”Other_Names”,

”Mothers_birth_name”, “Military_Rank”, “Ethnic_group”, “Comments”, “Telephone_Number”

R.E: Se anonimiza el Nombre, el Id, la fecha y hora de nacimiento, el sexo, otros nombres, el nombre de la madre, el Rango Militar, el Grupo étnico, los comentarios y el Número Telefónico del paciente. Se eliminan estos datos del DICOM a enviar.

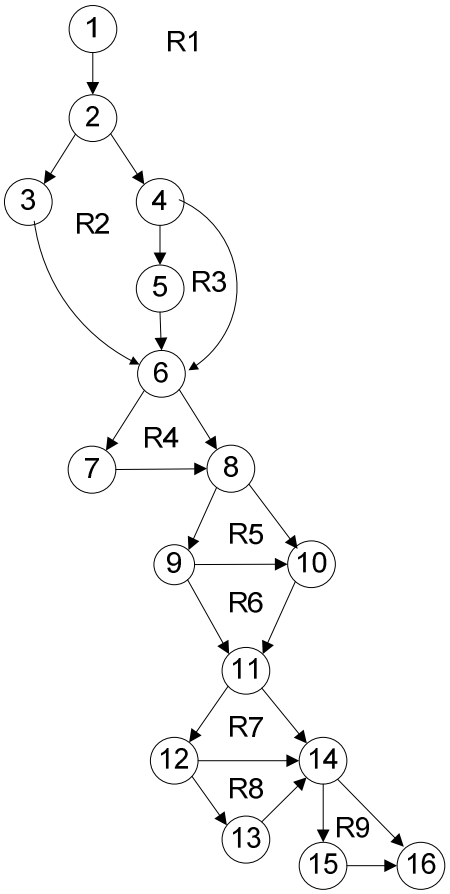
Caso 13: Anonymized Patient= “Name”,”Id”,”Birth_date”,”Birth_time”,”Sex”,”Other_Names”, ”Mothers_birth_name”, “Military_Rank”, “Ethnic_group”, “Comments”, “Telephone_Number”, “Preference_ Religious”

R.E: Se anonimiza el Nombre, el Id, la fecha y hora de nacimiento, el sexo, otros nombres, el nombre de la madre, el Rango Militar, el Grupo étnico, los comentarios, el Número Telefónico y la Preferencia Religiosa del paciente. Se eliminan estos datos del DICOM a enviar.

CB_Mod_DMail_Contacs Form

Void Validate TextBox

Grafo de Flujo



Complejidad Ciclomática

1ra Vía: V (G) = R9

2da Vía: V (G) = 23 – 16 + 2

V (G) = 9

3ra Vía: V (G) = 8 + 1

V (G) = 9

Caminos Básicos

Camino 1: 1-2-4-6-8-9-11-14-16

Camino 2: 1-2-3-6-8-9-11-14-16

Camino 3: 1-2-4-5-6-8-9-11-14-16

Camino 4: 1-2-4-6-7-8-9-11-14-16

Camino 5: 1-2-4-6-8-9-10-11-14-16

Camino 6: 1-2-4-6-8-10-11-12-14-16

Camino 7: 1-2-4-6-8-9-11-12-13-14-16

Camino 8: 1-2-4-6-8-9-11-13-14-16

Camino 9: 1-2-4-6-8-9-11-12-14-15-16

Casos de Prueba

Caso 1:

Name =”José Luis”

Alias =”Pepe”

Mail =”lleroz@uci.cu”

Phone =”58904379”

R.E: Se validan los datos introducidos en los campos y no se devuelve ninguna excepción de error.

Caso 2:

Name =” ”

Alias =”Tía”

Mail =”nidia@tranfins.cu”

Phone =”413893”

R.E: Se muestra un mensaje de error por no especificarse el nombre del contacto a adicionar.

Caso 3:

Name =”Yenni ??? Valdés”

Alias =”Ye”

Mail =”yvaldesh@estudiantes.uci.cu”

Phone =”8372635”

R.E: Se muestra un mensaje de error porque el nombre posee caracteres inválidos

Caso 4:

Name =”Yaimí Márquez”

Alias =” ”

Mail =”ymarqueza@estudiantes.uci.cu”

Phone =”8372635”

R.E: Se muestra un mensaje de error por no especificarse el alias del contacto a adicionar.

Caso 5:

Name =”Katia Hurtado”

Alias =”Katy”

Mail =”khurtado.uci.cu”

Phone =”8358130”

R.E: Se muestra un mensaje de error porque el correo insertado no es válido y las letras están de color rojo.

Caso 6:

Name =”Lourdes Escalona”

Alias =”Lourdita”

Mail =” “

Phone =”8358131”

R.E: Se muestra un mensaje de error porque el campo correspondiente al correo está vacío.

Caso 7:

Name =”Regla Díaz”

Alias =”Regla Softel”

Mail =”regla@softel.cu”

Phone =”835!??2”

R.E: Se muestra un mensaje de error porque el teléfono posee caracteres inválidos.

Caso 8:

Name =”Pedro Medina”

Alias =”Pedruco”

Mail =”medinariego@gmail.com”

Phone =” ”

R.E: Se muestra un mensaje de error por dejarse el campo correspondiente al teléfono en blanco.

Caso 9:

Name =”Juan José”

Alias =”Pepe”

Mail =”pepitin@yahoo.es”

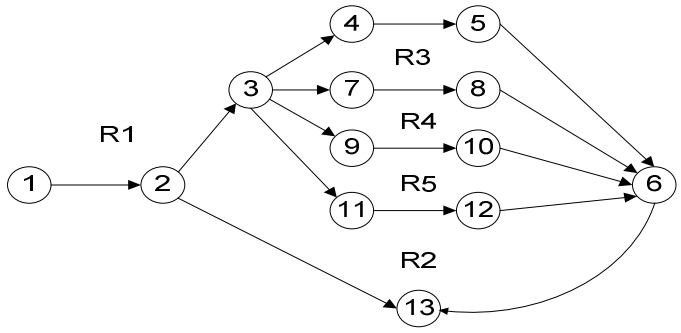
Phone =”514803”

R.E: Se muestra un mensaje de error porque el Alias se está tratando de duplicar, o sea, ya existe un contacto con ese Alias.

CB_Mod_DMail_ Contacts Form

Void Control_Key Down

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R5

2da vía: V (G)= 16 – 13 + 2

V (G)= 5

3ra vía: V (G)= 4 + 1

V (G)= 5

Caminos Básicos

Camino 1: 1-2-3-4-5-6-13

Camino 2: 1-2-3-7-8-6-13

Camino 3: 1-2-3-9-10-6-13

Camino 4: 1-2-3-11-12-6-13

Camino 5: 1-2-13

Casos de Prueba

Caso 1:

Keys = “Control + A”

R.E: Se modifica el mensaje a mostrar y se activa una nueva ventana de adicionar contacto.

Caso 2:

Keys = “Control + D”

R.E: Se modifica el mensaje a mostrar, se activa la excepción de eliminar contacto pidiendo confirmación para el caso que haya al menos uno seleccionado.

Caso 3:

Keys = “Control + S”

R.E: Se modifica el mensaje a mostrar, se activa una ventana de búsqueda para que el usuario inserte algún dato del contacto que desea localizar.

Caso 4:

Keys =”Control + E”

R.E: Se modifica el mensaje a mostrar, en el caso de estar seleccionado algún contacto se edita el mismo con todos sus datos correspondientes.

Caso 5:

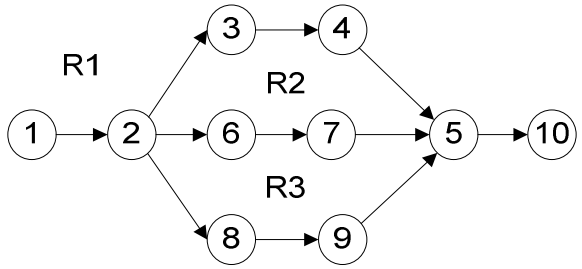
Keys! =”Control”

R.E: No se modifica el mensaje a mostrar.

CB_Mod_DMail_ Contacts Form

Void Editor_Key Down

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R3

2da vía: V (G)= 11 – 10 + 2

V (G)= 3

3ra vía: V (G)= 2 + 1

V (G)= 3

Caminos Básicos

Camino 1: 1-2-3-4-5-10

Camino 2: 1-2-6-7-5-10

Camino 3: 1-2-8-9-5-10

Casos de Prueba

Caso 1:

Keys = “Enter”

R.E: Se modifica el mensaje a mostrar. Se edita el contacto seleccionado con todos sus datos correspondientes si es que había alguno seleccionado.

Caso 2:

Keys = “Escape”

R.E: Se cierra el contacto que estaba abierto sin salvar los cambios en el mismo.

Caso 3:

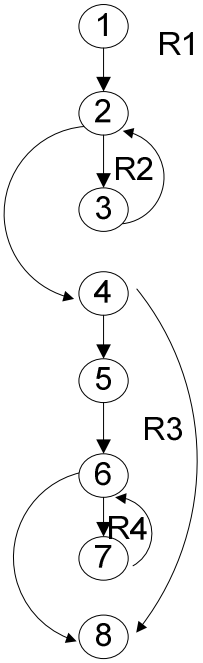
Keys = “Delete”

R.E: Se modifica el mensaje a mostrar. Se activa la excepción de eliminar contacto pidiendo confirmación para el caso que haya al menos uno seleccionado.

CB_Mod_DMail_ Contact Form

Void Edit_Click

Grafo de Flujo



Complejidad Ciclomática

1ra vía: V (G)= R4

2da vía: V (G)= 10 – 8 + 2

V (G)= 4

3ra vía: V (G)= 3 + 1

V (G)= 4

Caminos Básicos

Camino 1: 1-2-3-2-4-5-6-7-6-8

Camino 2: 1-2-3-2-4-8

Camino 3: 1-2-3-2-4-5-6-8

Camino 4: 1-2-4-8

Casos de Prueba

Caso 1:

Selected Rows! = 0

R.E: Se actualiza la información a mostrar, se edita el nuevo contacto con todos sus componentes.

Caso 2:

Selected Rows = 0

R.E: No ocurren cambios en la información a mostrar.

Caso 3:

Selected Rows! = 0

R.E: Se actualiza la información a mostrar, se edita el nuevo contacto.

Caso 4:

Selected Rows = 0

R.E: No ocurren cambios en la información a mostrar.

3.7 Bitácora de Prueba.

Id del Módulo	Errores	Tipo de Errores	Observaciones
CB_ModDmail	**Los errores más frecuentes son:** ✓ Algunas funciones no se activan y desactivan cuando debe ser. ✓ Hay mensajes que no emiten ningún texto aclaratorio.	Los errores que se pudieron encontrar son de tipo de formato.	✓El sistema esta muy bien estructurado en cuanto al orden de programación, solo quedan fragmentos de código innecesariamente comentariados

3.8 Resumen de los errores detectados.

El sistema no presenta ningún error de tipo catastrófico, solamente algunos problemas a la hora de especificar bien los mensajes de error que se le muestran al usuario. Por otra parte se le recomienda a los desarrolladores del mismo que en próximas versiones eliminen las líneas comentariadas innecesariamente. Tampoco se utiliza ningún estándar de codificación para la programación de este módulo Dmail por lo que se sugiere se tengan en cuenta estos aspectos también para una nueva mejora del mismo.

3.9 Comparación entre resultados esperados y Resultados Obtenidos.

Haciendo una breve comparación con los errores que se esperaban del sistema y los que se obtuvieron, podemos decir que el software realiza todas las funciones que tiene implementadas solo contiene algunas fallas de validación y de tratamiento de errores. También era probable que algunos eventos anidados emitieran fallas al restaurarse, y esto no ocurrió en ninguna ocasión, De forma genérica se puede plantear que se previeron la mayor parte de los resultados que generó el sistema, estando ya en proceso la codificación del mismo siguiendo estándares de codificación y sin los señalamientos encontrados.

3.10 Resultados de la Evaluación del Procedimiento para la realización de Pruebas de Caja Blanca usando la Técnica del Camino Básico.

La aplicación del Procedimiento Propuesto se evalúa de satisfactoria ya que durante su utilización para la revisión del Módulo Dmail del Sistema Cassandra PACS no hubo que buscar ningún otro documento o referencia a cerca de la técnica que se estaba empleando, pues todos los pasos estaban explícitos y correctamente descritos.

Las Pruebas realizadas al Código Fuente arrojaron algunos errores que aunque no se pueden clasificar como catastróficos continuaban vigentes en una aplicación que ya había sido probada con el Método de Caja Negra. Esto da una medida de la efectividad y confiabilidad que ofrece el Procedimiento empleado.

Por otra parte los Casos de Prueba así como los errores detectados quedaron registrados bajo un formato que posibilita tanto a probadores como implementadotes la fácil comprensión de los mismos, permitiéndose su reutilización ante cualquier modificación que sufra la aplicación sin necesidad de comenzar nuevamente desde cero.

3.11 Conclusiones

En el Capítulo 3 se expuso una breve descripción del Módulo DICOM Mail al cuál se le aplicó el Procedimiento de Pruebas Caja Blanca usando la Técnica del Camino Básico propuesto con anterioridad en este trabajo, con el objetivo de intentar descubrir la presencia de errores.

El uso del procedimiento propuesto facilitó el diseño de los casos de prueba, así como la ejecución de los mismo. Este procedimiento fue usado como una guía y aunque no se completaron todos sus modelos porque los diseñadores de prueba y probadores fueron las mismas personas, fue muy útil y garantizó que toda la revisión del código quedara formalmente documentada en un formato fácil de entender, incluso para el líder e implemetadores del sistema.

Conclusiones

A partir de toda la investigación realizada sobre la calidad de software y el proceso de prueba en la Facultad 7 de la Universidad de las Ciencias Informáticas se arribó a la conclusión de que la carencia de un procedimiento para la realización de las Pruebas de Caja Blanca, era uno de los grandes problemas que afectaban el buen funcionamiento de la revisión del código fuente, a los sistemas informáticos terminados.

Por esta razón, se propuso el diseño de un procedimiento para realizar dichas pruebas usando la Técnica del Camino Básico y cumpliendo con lo establecido en la Norma ISO 9000 del 2005.

Se determinó evaluar el procedimiento propuesto empleándolo para la revisión del Módulo Dmail del Sistema Cassandra PACS, obteniéndose resultados satisfactorios con su utilización.

Al finalizar el presente trabajo de diploma se dan por cumplidos los objetivos planteados, obteniéndose un Procedimiento que favorecerá el incremento de la eficiencia en la realización de las Pruebas de Caja Blanca en la Facultad 7 de la Universidad de las Ciencias Informáticas.

Recomendaciones

Una vez cumplidos los objetivos de este trabajo, y teniendo en cuenta las experiencias obtenidas durante la realización del mismo, se recomienda:

- ✓ Realizar el proceso de pruebas de software durante todas las fases de su desarrollo y no cuando ya estén terminados.

- ✓ Utilizar el Procedimiento propuesto de Pruebas de Caja Blanca usando la Técnica del Camino Básico para la revisión del código fuente de los sistemas informáticos en la Facultad 7 y hacerlo extensivo al resto de la Universidad en los casos que sea posible.

- ✓ Combinar esta técnica con la realización de Pruebas de Caja Negra y de esta forma lograr una revisión mucho más eficiente.

- ✓ Corregir lo más pronto posible los errores detectados en el *Modulo Dmail del Sistema Cassandra PACS.*

- ✓ Aplicar este procedimiento para la revisión del código fuente del resto de los módulos que componen el sistema Cassandra PACS.

- ✓ Automatizar el proceso de pruebas de caja blanca siguiendo lo establecido en el procedimiento.

- ✓ Utilizar los conceptos, casos ejemplos y resultados de este trabajo para lograr que los grupos de desarrollo tengan conocimiento de las técnicas de prueba y que las incluyan como una fase más dentro del desarrollo de los sistemas.

Referencias bibliográficas

- ✓ **[Cig_Labs, 2002]** "*The Home of Groundbreaking Software Quality*" Management Research, http://:www.cigitallabs.com/reso/definitions/software_testing.html, (08/02/2007).

- ✓ **[Fernández, 2002]** Fernández Peña J. M. IPN México. "*Pruebas de integración para componentes de software*", Marzo 2002.

- ✓ **[IEEE, 1991]** IEEE, IEEE Std1995, "*Metrics*", IEEE, 1991.

- ✓ **[Kan, 1995]** Kan, S.H. 1995: "*Metrics and models in software quality engineering*", AddisonWesley, Reading, Ma., USA, 1995.

- ✓ [McCabe, 1996] T.J.McCabe, "*Structured testing: a testing methodology using the cyclomatic complexity metric*", Technical Report NIST 500-225, 1996.

- ✓ **[Ma Isabel, 2001]** Alfonso María Isabel DCCIA, "*Tema 8. Pruebas del software.*"Universidad de Alicante, 2002.

- ✓ **[Meyer, 1997]** Meyer, B. 1997: "*Object oriented software construction*", 2nd. De., Upper Saddle River, Prentice Hall, 1997.

- ✓ **[MYE, 1979]** Myers G. "*The art of software testing*". Wiley. 1979.

- ✓ **[Pressman, 1997]** Pressman, R., "*Software Engineering: a practitioner's approach*", European Edition. McGraw Hill. 1997.

- ✓ **[Pressman, 1998]** Roger S. Pressman, R. "*Can Internetbased Applications Be Engineered?*" in IEEE Software, September/October IEEE Press, 104110, 1998.

- ✓ **[Pressman, 2000]** Roger S. Pressman: "*Software Engineering: A Practitioner's Approach*" (European Adaptation), McGrawHill. 2000. ISBN: 0077096770.

- ✓ **[Teruel, 2001]** Alejandro Teruel. http://www.ldc.usb.ve/~teruel/ci4713/clases2001/pruebasRep.html#bitacora. 25/04/2003).

- ✓ **[Yamaura, 1998**] Yamaura, Tsuneo 1998: "*How to design practical test cases*", IEEE Software v 15, n6, november/december 1998, pp 3036.

Bibliografía

- ✓ Alfonso Ma. Isabel http://www.dccia.ua.es/dccia/inf/asignaturas/IS2/teoria/tema7S.pdf (17/02/2007).
- ✓ Alfonso Ma. Isabel http://www.dccia.ua.es/dccia/inf/asignaturas/IS2/teoria/ejcT8S.pdf (15/04/2007).
- ✓ Alfonso Ma. Isabel http://www.dccia.ua.es/dccia/inf/asignaturas/IS2/teoria/tema8S.pdf (20/04/2007).
- ✓ Bashir I. and A.L. Goel 2000: "*Testing objectoriented software*". Springer, New York 2000.
- ✓ Beizer, Boris 1990: "*Software testing techniques",* 2nd edition*.* International Thomson Computer Press.
- ✓ Beizer, Boris 1995: "*Black box testing*". John Wiley and Sons, New York, 1995.
- ✓ Benson Ilke: 1999 http://www.ilkebenson.com/controldecalidad.htm (20/04/2007).
- ✓ Binder, R. V. 1999: "*Testing objectoriented systems. Models, patterns, and tools*".Addison Wesley Longman, Object Oriented Series, Reading Mass. 1999.
- ✓ Hernández J. F. and Minguet. M.J 2003: "*La Calidad del Software y su medida* ". Editorial CERASA (2003).
- ✓ Hunt, Neil. 1995: "*Testing objectoriented code*". Rational Software Tech. Papers. http://www.rational.com/support/techpapers/boundary.html (10/04/2007).
- ✓ Kan, S.H. 1995: "*Metrics and models in software quality engineering*", AddisonWesley, Reading, Ma., USA, 1995.
- ✓ **McCabe, T.J**. "*Complexity Measure*", IEEE Transactions on Software Engineering, vol. 2, n. 4, pp 308-320, December 1976.
- ✓ Medina Verónica http://itzamna.uam.mx/alfonso/pacs.html (26/04/2007)
- ✓ Perry, D.E. and G.E. Kaiser 1990: "*Adequate testing and objectoriented programming*", J. Objectoriented programming, Vol. 2, Jan/Feb 1990, pp 1319.
- ✓ Piattini F and Garcia E 2002: "*Calidad en el desarrollo y mantenimiento del Software*". Editorial RA-MA, Madrid, 2002.

Glosario de términos

Sistemas PACS

PACS son las siglas anglosajonas **P**icture **A**rchiving and **C**ommunication **S**ystem (Sistema de archivo y transmisión de imágenes). Se trata de un sistema computarizado para el archivo digital de imágenes médicas y para la transmisión de estas a estaciones de visualización dedicadas a través de una red informática.

DICOM

DICOM (**D**igital **I**maging and **Co**mmunication in **M**edicine) es el estándar reconocido mundialmente para el intercambio de imágenes médicas, pensado para el manejo, almacenamiento, impresión y transmisión de imágenes médicas. Incluye la definición de un formato de fichero y de un protocolo de comunicación de red. Los ficheros DICOM pueden intercambiarse entre dos entidades que tengan capacidad de recibir imágenes y datos de pacientes en formato DICOM.

Calidad.

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

Control de calidad.

Conjunto de técnicas y actividades de carácter operativo, utilizadas para verificar los requerimientos relativos a la calidad del producto o servicio.

El control de calidad (QC = *quality control*) es un componente esencial de un programa de digitalización de imágenes, y tiene como fin asegurar que se han cumplido las expectativas en cuanto a calidad. El mismo abarca procedimientos y técnicas para verificar la calidad, precisión y consistencia de los productos digitales.

Gestión de calidad.

Aspecto de la función de gestión que determina y aplica la política de la calidad, los objetivos y las responsabilidades y que lo realiza con medios tales como la planificación de la calidad, la garantía de calidad y la mejora de la calidad.

Anexos

Anexo 1: Formato usado para definir el Procedimiento.

UCI Universidad de las Ciencias Informáticas	**PROCEDIMIENTO PARA LA REALIZACIÓN DE LAS PRUEBAS DE CAJA BLANCA USANDO LA TÉCNICA DEL CAMINO BÁSICO**

Elaborado por: Firma: Yaimí Márquez Alpízar Yenni Valdés Hechavarría Fecha:	Revisado: Fecha:	Aprobado: Fecha:	Pág. 105 de 130

Universidad de las Ciencias Informáticas	**PROCEDIMIENTO PARA LA REALIZACIÓN DE LAS PRUEBAS DE CAJA BLANCA USANDO LA TÉCNICA DEL CAMINO BÁSICO**

1. Introducción

1.1 Objetivo

1.2 Alcance

1.3 Definiciones, Acrónimos y Abreviaturas

2 Desarrollo

2.1 Roles y Responsabilidades.

2.1 Introducción a la Técnica del Camino Básico.

2.1Cronograma de Actividades.

3. Documentos de referencia.

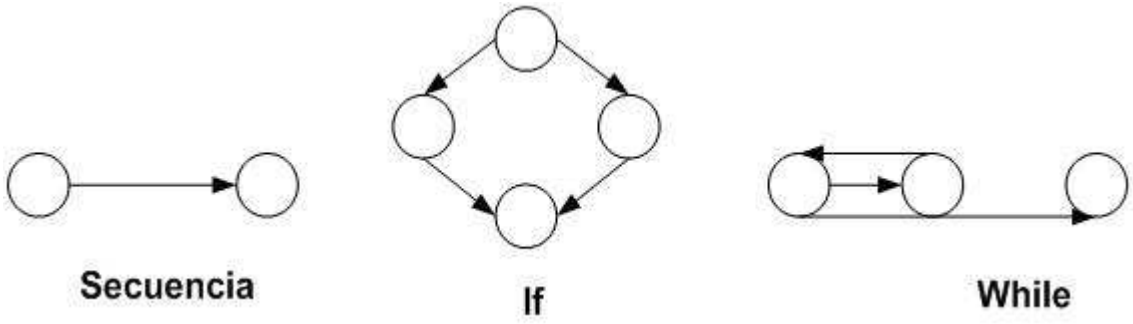
4. Modelos.

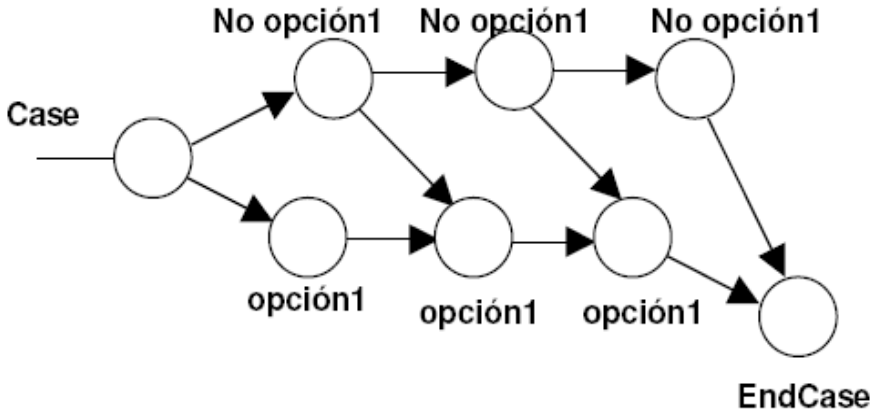
5. Distribución y Archivo.

6. Anexos.

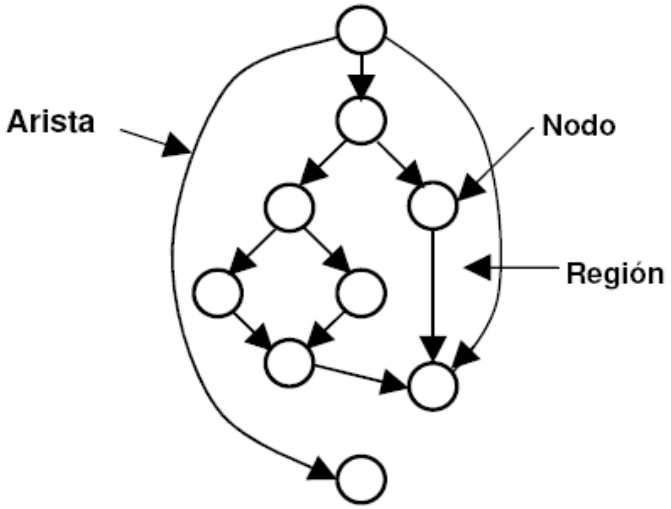
Elaborado por: Firma: Yaimí Márquez Alpízar Yenni Valdés Hechavarría Fecha:	Revisado: Fecha:	Aprobado: Fecha:	Pág. 106 de 130

Anexo 2: Notación de grafos de flujo para las diferentes instrucciones

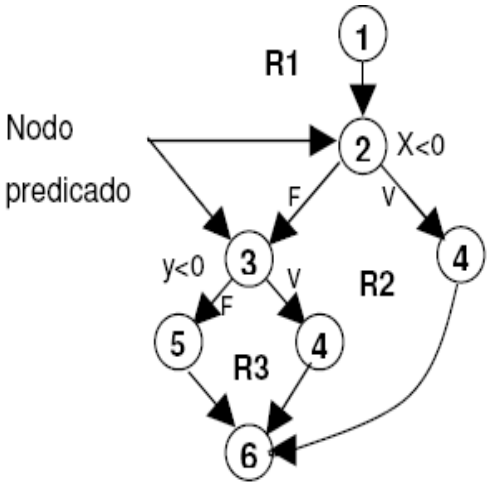




Anexo 3: Características de los Grafos de Flujo



Anexo 4: Ejemplo de un Grafo de Flujo




```
PROCEDURE Imprime _ media(VARx, y : real;)
VAR resultado : real;
resultado:=0;
IF (x < 0 ORy < 0) THEN
        WRITELN("x e y deben ser positivos");
ELSE
```

Anexo 5: Plantilla para Especificación de Casos de Prueba de Caja Blanca.

TITULO : Especificación de Casos de Prueba

[Nota: La siguiente plantilla se ha desarrollado para su uso en el Proceso de pruebas de software. El texto que se encuentra entre corchetes y presentado en estilo itálica azul se ha incluido para proporcionar una guía para el autor y se debería borrar antes de la entrega del documento.]
[Hay que sustituir el texto resaltado con marcador amarillo por su equivalente en el proyecto de desarrollo y eliminar el resaltado]

Nombre del Proyecto: *[Nombre del Proyecto]*

COPIA CONTROLADA N^{O} : 1

ASIGNADA A : Facultad 7 de la Universidad de las Ciencias Informáticas

REDACTADO POR : *[*Nombre y Firma del Diseñador de Prueba y del probador*]*	*[*Fecha*]*
REVISADO POR : *[*Nombre y firma del líder del Laboratorio de Calidad*]*	*[*Fecha*]*
APROBADO POR: *[*Nombre y Firma del líder del Laboratorio de Calidad*]*	*[*Fecha*]*

INDICE

1. Nombre del Caso de prueba.

<Nombre del caso de prueba>
[Para el nombre del caso de prueba se usa el formato CB por ser pruebas de Caja Blanca, a continuación el nombre del modulo que se esta probando seguido de la forma y el método que se esta revisando
Ejemplo: ***CB_Mod Dmail_*** **Message Form**
Void SendMail
Aquí se esta probando en el módulo DMail la función Enviar Mensaje de la forma Mensaje]

[

1. *Las variantes de las pruebas que se realizarán depende de las diferentes entradas que tenga el caso de prueba]*

2. <Primera variante del caso de prueba>

2.1. Descripción

[Breve descripción del desarrollo y objetivos del caso de prueba para esta variante. Ejemplo es necesario determinar si el mensaje fue enviado correctamente con todos los archivos adjuntos]

2.2. Condiciones de ejecución

[Descripción de las condiciones de ejecución que se deben cumplir antes de iniciar el caso de prueba por ejemplo, la conexión debe estar previamente establecida con el destinatario.]

2.3. Entrada

[Descripción paso a paso de la ejecución del caso de prueba]

-
-
-

2.4. Resultados Esperados

[Descripción del resultado que se esperaba del sistema una vez entrada esta variante]

2.5. Evaluación de la Prueba

[Aquí se deben poner los resultados obtenidos con la prueba.]

3. **<Otra variante del caso de prueba>**

3.1. Descripción

3.2. Condiciones de ejecución

3.3. Entrada

3.4. Resultados Esperados

3.5. Evaluación de la Prueba

4. <Variante n del caso de prueba>

4.1. Descripción

4.2. Condiciones de ejecución

4.3. Entrada

4.4. Resultados Esperados

4.5. Evaluación de la Prueba

Anexo 6: Bitácora de Pruebas

[Nota: La siguiente plantilla se ha desarrollado para su uso en la facultad 7. El texto que se encuentra entre corchetes y presentado en estilo itálica azul se ha incluido para proporcionar una guía para el autor y se debería borrar antes de la entrega del documento.] *[Hay que sustituir el texto resaltado con marcador amarillo por su equivalente en el proyecto de desarrollo y eliminar el resaltado]* *:[Nombre del Proyecto]* *:[Nombre de los Módulos Revisados]* *:[Lenguajes de programación en los que esta implementado]*	
REDACTADO POR : *[*Nombre y Firma del probador*]*	*[*Fecha*]*
REVISADO POR : *[*Nombre y firma del líder del Laboratorio de Calidad*]*	*[*Fecha*]*
APROBADO POR: *[*Nombre y Firma del líder del Laboratorio de Calidad*]*	*[*Fecha*]*

INDICE

1. Ambiente de Ejecución

[especificación de las condiciones de hardware y software bajo las cuales se ejecutaron las pruebas]

[especificación de la cantidad de computadoras que se utilizaron para las pruebas, bajo que condiciones, que tiempo duraron las mismas y cuantas personas intervinieron en estas]

Tablas de Corridas

Id del Módulo	Errores	Tipo de Errores	Observaciones
[se usa el formato CB por ser pruebas de Caja Blanca, a continuación el nombre del modulo que se esta probando Ejemplo: *CB_ModDmail*	*[La descripción de los errores debe ser lo más precisa y completa posible.* *Para facilitar el seguimiento posterior del error cada uno debe recibir un identificador único.]*	*[Los errores se pueden clasificar según el Tipo que haya arrojado el sistema:* ***Catastrófico**: el caso no logra producir salidas porque el ambiente de pruebas se cae o porque el software bajo prueba termina anormalmente sin producir las salidas o sin producirlas completas.* ***Funcional:** los valores de las salidas de la ejecución discrepan de los valores esperados.* ***Propiedad (se especifica también la propiedad):** se incumple una propiedad. Por ejemplo un error debido a que el tiempo de respuesta del sistema es mayor a lo exigido, es una falla de propiedad (desempeño).* ***Formato:** la salida no cumple con el formato esperado para ella, pero sus valores sí son los esperados.* *La tipificación del error siempre se hace en la categoría más alta que puede alcanzar.* *Así, un error que produce sólo una de sus dos salidas, y la que sale tiene el valor equivocado, es catastrófico.* *Similarmente uno que produce todas sus salidas, tiene una equivocación en una de sus salidas y además no cumple con el tiempo de respuesta*	*[se especifican los detalles de los errores.* *se ponen otros señalamientos que no sean directamente errores en la codificación, como por ejemplo si los mensajes mostrados son suficientes para guiar al usuario]*

		que se le exige, se clasifica como un error funcional (recuerde que una respuesta equivocada puede obtenerse tan rápido como se quiera...). *Un error que incumple una propiedad y un formato, se considera de propiedad. En ningún caso, la categorización* *exime de la necesidad de describir todas las características del error.]*	

2.1 <Tabla de Corrida para el Módulo DMail>

Id del Módulo	Errores	Tipo de Errores	Observaciones

2.2 <Tabla de Corrida para el Módulo n>

Id del Módulo	Errores	Tipo de Errores	Observaciones

3. Anomalías.

[eventos o incidentes no previstos, desviaciones de los procedimientos previstos, prueba no completada]

3.1. Descripción

[Descripción de las causas que ocasionaron 3.]

4. Resumen de los errores detectados.

[se hace un resumen general de todos los errores encontrados en el sistema revisado]

4.1. <Resumen de errores detectados en el Módulo DMail.>

4.2. <Resumen de errores detectados en el Módulo *n.*>

5. Comparación entre resultados obtenido y esperados.

Debe ser breve y lo mas general posible sin entrar en detalles de cada módulo, en forma de párrafo o de tabla, destacando los aspectos mas relevantes.

Anexo 7: Acta de Entrega de Evaluación

<LOGO del producto>
<Nombre del Producto>
Acta de Entrega

Versión <1.0>

[Esta plantilla se ha confeccionado como guía para la entrega de todos aquellos productos tangibles e intangibles necesarios para su evaluación.]

[Esta plantilla es aplicable para todos aquellos clientes que hayan terminado su producto y necesiten una evaluación y revisión del código fuente, el acta se confeccionará para la entrega legal y confidencial al equipo de pruebas de cada una de las partes que componen el producto. Además en caso de que se rechace el producto la presente acta quedará como crédito de la devolución al cliente de lo que antes le fue entregado.]

[Esta plantilla es la constancia legal de todo lo que se entrega para respaldar el producto (documentos, código etc.), que tiene que ser bajo un compromiso legal de confidencialidad y confiabilidad, es de obligatorio cumplimientos para todas aquellas partes que reciban el producto. En este documento solo deben ser marcados los elementos que han sido entregados al equipo de pruebas. En su segunda parte este documento es legal para la devolución de lo que antes le ha sido confiado al equipo.]

TABLA DE CONTENIDOS

ACTA DE ENTREGA

1. INTRODUCCIÓN

[En esta plantilla aparece todo lo posible a entregar después de la realización y terminación de un producto competitivo, deben ser marcados los documentos y las demás partes que componen el producto y especificadas las cantidades que fueron entregadas y de esta misma forma debe hacerse en el momento que el cliente reciba el software en caso de rechazar su solicitud de certificación.]

1.1 PROPÓSITO Y OBJETIVOS

[El objetivo de este documento es que quede una constancia legal de la entrega y devolución íntegra del producto.]

1.2 ALCANCE

[Esta plantilla es necesaria para todos los clientes que hayan solicitado la revisión del código fuente de su software y que hayan cumplido con las reglas que se definen y establecen en el Expediente del Producto]

1.3 DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

[En caso de usarse en la plantilla términos de difícil comprensión o siglas no evidentes]

1.4 REFERENCIAS

[Se colocan las referencias necesarias sobre documentos utilizados o necesarios para la comprensión de la presente acta de entrega]

1.5 RESUMEN

[Resuma los elementos que considere fundamentales a tener en cuenta para la confección de su acta de entrega, destacando de manera general los escogidos a entregar y cualquier detalle que considere necesario y no esté implícito en la actual planilla]

2 DEFINICIÓN DE RESPONSABILIDADES

*[**Cliente**: representante de los desarrolladores.*

***Jefe del Equipo de Pruebas**: representante de los probadores.]*

3 CUERPO DE LA PLANTILLA

Marque con una (X) los componentes que conforman su producto, para el resto No Procede (NP) y si es necesario especifique la cantidad, unidades de medidas etc.

a) ____ Descripción del Producto
b) ____ Requisitos de hardware
c) ____ Requisitos de software
d) ____ Requisitos de documentación
e) ____ Requisitos de personal
f) ____ Otros Requisitos
g) ____ Documentación para el usuario
h) ____ Documentación del paquete
i) ____ Caso de ensayo /prueba, (especificar la cantidad)

[Instrucción documentada para la persona que realiza los ensayos /pruebas que especifican cómo se tiene o tendría que probar una función o una combinación de funciones. Un caso de ensayo /prueba incluye información detallada sobre los siguientes pasos:

- *el objetivo del ensayo / prueba;*
- *las funciones que se han de comprobar;*
- *el entorno del ensayo / prueba y otras condiciones (detalles de configuración y trabajos preparatorios);*
- *los datos del ensayo/prueba;*
- *el procedimiento;*
- *el comportamiento esperado del sistema.]*

j) ____ Programas y datos necesarios
k) ____ Componentes del sistema

[Deben estar presentes todos los elementos que se mencionan en la descripción del producto.]

l) ____ Preparación del plan de formación

[Sobre la base de los materiales para el uso del producto y de la propia instalación, debe ser necesario tanto para el cliente como para la evaluación por cualquier tercero]

Afirmaciones referentes a confiabilidad:

Yo _____ Líder del equipo de Pruebas declaro que una vez entregado el producto a evaluar es de sumo compromiso legal la confidencialidad y seguridad de esta información entregada.

el día____ del mes_____ del año _____.

Nombre y Firma del cliente (entrega):	**Nombre y Firma Líder del Equipo de pruebas (recibe):**
_____	_____

Firma legal de Recibimiento del expediente por parte del cliente

[Esta sección de la planilla sólo se llenará en caso de rechazar la solicitud de certificación hecha por el cliente por cualquiera de las razones posibles]

Lista de las no conformidades con la entrega

[Liste en este apartado de forma detallada los elementos que no le fueron entregados, tomando como referencia el listado anteriormente firmado por usted]

[Coloque cualquier no-conformidad en la forma, tratamiento o uso inadecuado de los elementos entregados al equipo de pruebas]

Elemento entregado	Cantidad	Detalles de la No-conformidad	Otros comentarios
[Haga referencia al índice del elemento. Ejemplo: k)]			

Afirmaciones referentes a confiabilidad:
Yo _____ representante de los desarrolladores del software declaro que el día____ del mes_____ del año _____ me fueron entregados todos los medios que puse a disposición del Equipo de Pruebas de Caja Blanca.

Nombre y Firma del cliente

(recibe):

_____

Nombre y Firma Líder del equipo de pruebas
(entrega):

_____

Anexo 8: Encuesta

De antemano le agradecemos su colaboración y el tiempo que ha dedicado a la misma. Gracias.

1. Su conocimiento acerca de la calidad de software lo puede catalogar como:

☐Mucho ☐Regular ☐Poco ☐Nada

2. Un elemento crítico para determinar la calidad del software son las pruebas que se le realizan al mismo. ¿Cuánto conoce Ud. acerca del tema?

☐Mucho ☐Regular ☐Poco ☐Nada

3. ¿Le ha realizado pruebas a algún software?

☐Sí ☐No

En caso afirmativo, diga:

a. ¿Qué tipos de pruebas ha aplicado?

☐Prueba de Caja Negra.

☐Prueba de Caja Blanca.

☐Prueba a la documentación.

☐Otras. ¿Cuales?

_____
_____

4. ¿Cómo ve Ud. la aplicación de las pruebas de software en su proyecto?

☐Adecuada ☐Media ☐Insuficiente

5. En caso de que considere que no sea adecuado el proceso de pruebas, ¿cuáles son, a su entender, las causas por la que esto ocurre?

☐Falta de una adecuada estructura organizativa para este efecto.

☐Desconocimiento del tema o técnicas a aplicar.

☐Escaso personal capacitado y disponible para ello.

☐Otros_____
_____

6. ¿Qué importancia le concede Ud. a la aplicación de las pruebas del software en su proyecto?

☐Mucha ☐Regular ☐Poca ☐Ninguna

7. Exprese brevemente algunas sugerencias que Ud. tendría en cuenta para lograr mejorar la calidad de los softwares en nuestra universidad.

_____
_____
_____
_____
_____