UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 9

TÍTULO: **Diseño y aplicación de pruebas al producto Captura y Catalogación de Medias (CCM).**

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

AUTOR: **Gilberto Velázquez Rodríguez**

TUTOR: **Ing. Anay Iyenis Chapman Hernández.**

Ciudad Habana junio 2010

DEDICATORIA

Este trabajo se lo dedico especialmente a la memoria de mi madre Xiomara Rodríguez González, que aunque la vida me separo de ella en edades muy temprana de mi vida no ha sido motivo suficiente para que ella permanezca en mi corazón y mi mente. Siendo ella mi fuente de inspiración en todo momento de mi vida.

A mi padre por darme este tamaño llenándome de amor y apoyándome en mis decisiones, confiando en mí en todo momento y por guiarme durante toda mi vida, forjándome en el hombre que soy hoy.

A tres Ángeles que la vida me dio cuando la ausencia de mi madre invadió mi ser, ellas son mis tías, Sonia, Zeida y Gisela por darme su amor incondicional, apoyándome en todo momento y darme las fuerzas necesarias para continuar siempre mis estudios.

AGRADECIMIENTOS

A mi madrastra por su ayuda incondicional, confiar en mí en todo momento y tratarme como un hijo más.

A mis tíos, Gustavo y Arcel por estar pendiente de mis estudios y guiarme en todo momento, aconsejándome y apoyándome en cada decisión tomada.

A mi Abuela Ofelia por quererme tanto y cuidarme en los momentos más difíciles de mi vida.

A mi hermana Marlenis por estar pendientes de mis problemas y brindarme todo su amor y su cariño.

A Tristá por ayudarme todo este tiempo, compartiendo en todo momento y tratándome como un hijo más.

A mi tía Claribel por quererme de una forma transparente ya apoyarme tanto en mis estudios.

A Lianet y Baloe por tratarme como un hermano y guiarme en vida profesional con sus consejos.

A mis primos Isaac, Gustavito, Yary, Luis y Iry por quererme tanto y apoyarme siempre.

A mi tutora Anay por apoyarme todo el tiempo durante este trabajo, guiándome y dándome fuerzas, por sus críticas y su dedicación hacia mí.

A mi tribunal por su ayuda, consejos y seguimiento con el trabajo.

A mis amigos que conocí en la Universidad que siempre voy a recordar, por su ayuda en cada momento de mi carrera.

Agradezco a la Revolución y en especial a Fidel por darme la oportunidad de estudiar en esta universidad.

En fin a todas aquellas personas que de una forma u otra que me brindaron su ayuda…

DECLARACIÓN DE AUTORÍA

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

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

Autor: Gilberto Velázquez Rodríguez

_____

Tutor: Ing. Anay Iyenis Chapman Hernández.

_____

RESUMEN

En la Universidad de las Ciencias Informáticas (UCI), específicamente en el Departamento de Señales Digitales, surge la necesidad de la construcción de un producto informático, dedicado a la captura y catalogación de medias. En el mundo, estos sistemas son realizados por prestigiosas compañías del software, que cuentan con las herramientas y equipos especializados dedicados a garantizar la calidad de sus productos.

El presente trabajo, titulado “Diseño y aplicación de pruebas al producto Captura y Catalogación de Medias (CCM)” se realizó con objetivo fundamental de garantizar la calidad a este producto, proporcionando así una buena aceptación por parte del cliente. Para dar cumplimiento a este objetivo se hizo indispensable llevar a cabo todo un proceso de pruebas, con el fin de detectar la mayor cantidad de defectos; y una vez identificados estos, corregirlos en el menor tiempo posible.

Para ello fue necesario realizar una fundamentación teórica en la que se abordan los diferentes conceptos emitidos por varios autores referentes a aspectos relacionados con la calidad de software, las pruebas de software y el aseguramiento de la calidad en el proceso de desarrollo.

Se examinó todo lo referente al flujo de pruebas que define RUP (Rational Unified Process) como metodología de desarrollo de software, la cual se escogió por el equipo de desarrollo para guiar el proceso de construcción del mismo. Analizando también los tipos de pruebas y los niveles a los cuales se aplican los mismos, logrando con esto, complementar el grado de calidad de un producto informático.

Se mencionan las principales características de cada módulo del proyecto CCM, favoreciendo con esto comprobar cada funcionalidad del producto. Por otra parte se confeccionó un plan de pruebas que recoge paso a paso todas las características a seguir en el diseño y aplicación de los casos de pruebas al producto CCM. Se utilizaron las técnicas de caja negra y caja blanca para la realización de las pruebas, lo que permitió la detección de un gran por ciento de errores. Facilitando la documentación de estos, para que fuesen corregidos de una mejor forma por el equipo de desarrollo. También se lleva a cabo una comparación entre los resultados obtenidos con los esperados, documentando recomendaciones para posteriores versiones del producto.

PALABRAS CLAVES

Calidad, estrategia, pruebas software

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

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

Uno de los usos más amplios que se le ha dado a los avances tecnológicos ha sido apoyar y beneficiar el procesamiento automático de información, conocido mundialmente con el término informática. En sus inicios esta rama era completamente artesanal, producto a la desorganización y a la falta de planificación. Sin embargo, en la actualidad se dedican esfuerzos a mejorar lo que se hizo mal en el pasado, lo cual ha traído consigo que se logren avances significativos en esta esfera, con un alto grado de calidad en los procesos que desencadenan la creación de un producto informático.

Cuba no está exenta de este progreso que ha tenido las tecnologías de la información y las comunicaciones (TIC), por tal motivo la dirección del estado cubano, consciente de la facilidad que proporcionan las TIC para elevar el avance cultural de la sociedad en general, no solo prioriza y promueve su informatización, para lo cual dedica innumerables esfuerzos; sino que aboga por el desarrollo de soluciones informáticas encaminadas a solucionar diversos problemas.

En vista de las exigencias de la industria del software de hoy día, ha sido necesario y de suma importancia hacer programas que posean una buena calidad, para poder entrar en el mercado tanto nacional como internacional. El incremento de la complejidad de los productos informáticos aumenta; unido a esto crece de forma acelerada la necesidad de asegurar la calidad de los mismos.

Como parte de las TIC, las telecomunicaciones[1], se han convertido en uno de los sucesos más importantes de las últimas décadas. Su objetivo principal es mejorar las condiciones de trabajo y de vida de los seres humanos, garantizando una mayor difusión de la información. Dentro de la misma se pueden encontrar variadas formas de comunicación a distancia, destacándose de estas la televisión.

La televisión ha experimentado un auge tecnológico, en la actualidad. Insertándose en las más disímiles prácticas populares, por lo que constituye un medio del cual pocas personas pueden prescindir. Debido a la alta demanda que tienen estos medios audiovisuales como parte de la

[1] *Es una técnica consistente en transmitir un mensaje desde un punto a otro, normalmente con el atributo típico adicional de ser bidireccional. El término telecomunicación cubre todas las formas de comunicación a distancia, incluyendo radio, telegrafía, televisión, telefonía, transmisión de datos e interconexión de ordenadores a nivel de enlace.*

sociedad, en todo el mundo ha aumentado la creación de productos informáticos destinados a facilitar el trabajo relacionado con los mismos.

La Universidad de la Ciencias Informáticas (UCI) primera universidad surgida en Cuba al calor de la batalla de ideas, constituye un pilar fundamental para desarrollar la informática en el país. Haciendo uso de las bondades de las TIC, se desarrollan en la institución soluciones integrales de software y se brindan servicios. El objetivo principal de la UCI es desarrollar productos informáticos para satisfacer la demanda de Cuba o para que se puedan insertar en el mercado internacional.

Para dar cumplimiento a este objetivo la UCI cuenta con una serie de centros de producción en los cuales se desarrollan productos informáticos de disimiles ramas. En la facultad 9 de la misma está el centro Sistema Digitales que cuenta con 2 departamentos. Uno de ellos es el departamento de Señales Digitales, en el cual se desarrollan los temas afines con la televisión digital y el procesamiento de imágenes. En el mismo se está implementando un sistema para la Captura y Catalogación de Medias (CCM).

En el mundo los sistemas dedicados a estos fines están realizados por prestigiosas organizaciones, que tienen amplia experiencia en esta rama. Los mismos cuentan con herramientas y equipos especializados para la fabricación con calidad de sus productos. Por tal motivo la realización de este sistema informático implica un gran compromiso por parte de cada uno de los miembros del equipo de desarrollo.

Es un reto para la UCI competir con otros sistemas de captura, catalogación y monitoreo de televisión desarrollados por grandes compañías de software a escala internacional. Por lo que se convierte en una necesidad concluir este sistema con todas las exigencias pertinentes, para que supere en cierta medida las características de los que actualmente se comercializan a escala mundial.

Teniendo en cuenta lo planteado anteriormente se hace imprescindible en todo el proceso de desarrollo de software tomar las medidas pertinentes por todos los integrantes del equipo para que el producto deseado cumpla las expectativas esperadas. Es de suma importancia tener un control de cada una de las etapas de desarrollo, con mecanismos fiables que posibiliten la detección de posibles errores que puedan cometer los miembros del equipo de trabajo. Lo que posibilitaría tener un registro de estos. Minimizando así las posibles consecuencias negativas que se pudieran presentar si no se corrigen en el tiempo adecuado. Entre estas se pudieran citar: costo, tiempo del ciclo de desarrollo del producto.

Por ello se ha convertido en una necesidad primordial, desarrollar un proceso de pruebas para mitigar las situaciones de discrepancias entre lo que debía ser y lo que se realizó realmente, antes de poner en uso el producto por los usuarios finales. Por lo que el **problema a resolver** consiste en: ¿Cómo

garantizar que los niveles de calidad para el producto Captura y Catalogación de Medias (CCM) cumpla con los requisitos de futuros clientes?

Para validar la calidad de este producto se propone realizar un adecuado diseño de pruebas. Por tanto el **objeto de estudio** de la investigación es: el proceso de pruebas como parte del desarrollo de software.; enmarcando el **campo de acción**: proceso de pruebas de calidad del software aplicado al producto Captura y Catalogación de Medias.

Basado en lo anteriormente mencionado, el **objetivo general** de este trabajo es: diseñar y aplicar casos de prueba al producto Captura y Catalogación de Medias (CCM) para garantizar que los niveles de calidad del mismo cumplan con los requisitos del cliente.

Para el cumplimiento del objetivo general se definieron diversas **tareas de investigación**, las cuales son:

1. Evaluar el estado del arte de las estrategias de pruebas a utilizar en proyectos informáticos; y los principales conceptos y herramientas relacionados con la calidad de software.
2. Seleccionar la estrategia de prueba para el producto (CCM).
3. Seleccionar los casos de uso del producto Captura y Catalogación de Medias (CCM) a los cuales se les va a realizar el diseño de casos de pruebas.
4. Diseñar casos de pruebas a cada uno de los casos de uso seleccionados del producto CCM.
5. Definir los casos de uso que serán probados del producto CCM para la estructuración de la ejecución de las pruebas.
6. Aplicar las pruebas a los casos de uso del producto CCM que fueron seleccionados con anterioridad.
7. Evaluar los resultados obtenidos de la aplicación de las pruebas.

Por todo esto se tiene como **hipótesis** lo siguiente**:** si se realiza adecuadamente el diseño de casos de prueba y se aplican estos de forma efectiva al producto Captura y Catalogación de Medias (CCM), se logrará un producto final con la calidad requerida que cumpla con los requisitos del cliente.

Una vez concluida la investigación se espera obtener los siguientes resultados:

- ✓ Diseño de Casos de Pruebas del producto CCM.
- ✓ Plan de Pruebas del producto CCM.
- ✓ Evaluación de las pruebas realizadas a los casos de uso del producto CCM.

Con el propósito de darle solución al objetivo trazado durante la investigación, se utilizaron varios métodos científicos:

Métodos teóricos: con la utilización de estos es posible estudiar las características del flujo de trabajo de prueba de un software que no son observadas directamente.

- ✓ **Método histórico-lógico:** se utilizó el mismo para investigar la información que se tiene hasta el momento del estado del arte, sobre las estrategias de pruebas utilizadas en los proyectos informáticos; y los principales conceptos y herramientas relacionados con la calidad de software.
- ✓ **Método analítico sintético:** este método se aplicó con el objetivo de realizar un estudio detallado de los casos de uso del producto Captura y Catalogación de Medias (CCM) con el fin de seleccionar los casos de uso a los que se le van a realizar las pruebas.
- ✓ **Método de modelación:** este método se usó para diseñar casos de pruebas a cada uno de los casos de uso estudiados del producto (CCM). Pues se convierte en una necesidad que se ejecute esta modelación a la hora de realizar el diseño a cada uno de los casos de usos que se escogieron para aplicar las pruebas correspondientes.

Métodos empíricos: estos métodos son los que permiten hacer una retroalimentación de trabajos realizados con anterioridad sobre un tema en especifico, en este caso para el diseño y aplicación de pruebas al producto CCM y mediante estos se pueden analizar diferentes variantes.

- ✓ **Método de Observación:** este se utilizó en toda la investigación, pues se recoge información de los aspectos tratados en la tesis desde el punto de vista de otros autores así como sus definiciones y resultados para permitir realizar un análisis del arte en ese campo.
- ✓ **Método de Entrevista:** Este método se utilizó con el objetivo fundamental de seleccionar aquellos casos de usos que iban a ser sometidos a pruebas.

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

Capítulo 1. En este capítulo se realizó un análisis de todos los aspectos fundamentales que están relacionados con las pruebas de software. Se hace referencia a la metodología de desarrollo por la cual se guiará todo el proceso durante la fase de pruebas. Se hace referencia a lo que es un plan de pruebas, así como los modelos de calidad existentes. Además se trataron los principales aspectos relacionados con los tipos de pruebas, niveles de los mismos así como las técnicas de pruebas más utilizadas en los productos informáticos. En este capítulo se demostró la importancia que tiene el proceso de pruebas durante el proceso de desarrollo de un producto informático, así como las consecuencias que trae si no se lleva a cabo el mismo.

Capítulo 2. En este capítulo se realizó todo el diseño de caso de pruebas a los casos de uso seleccionados para ser objetos de prueba, al producto generado por el proyecto Captura y

Catalogación de Medias (CCM). Se realizó el plan de pruebas donde queda plasmada toda la estrategia a seguir, para que las pruebas sean llevadas a cabo eficientemente. Se especifica la configuración del entorno en que serán realizadas. Además se hace referencia a los procedimientos utilizados para su desarrollo, que sirvieron de base para el diseño y la ejecución de los casos de pruebas.

Capítulo 3. En este capítulo se realizaron todas las pruebas establecidas en el capítulo anterior. Además se realizó un análisis de los resultados que se obtuvieron durante todo el proceso de pruebas que se llevó a cabo sobre el producto generado por el proyecto CCM. Se realizó un comparación entre los resultados que se esperaban obtener del producto CCM con los resultados que se obtuvieron realmente, logrando con esto tener una visión más profunda de la calidad que tiene en la actualidad este producto. Se redactó un resumen donde están plasmadas las principales deficiencias encontradas durante todo el proceso y se plasmaron las principales recomendaciones a tener en cuenta para posteriores trabajos.

CAPÍTULO 1

Fundamentación Teórica.

1.1. Introducción

Durante el proceso de desarrollo de software la etapa de pruebas constituye una de las fases más importantes del ciclo de vida, pues con ésta se logra depurar un software desde sus inicios, para que el mismo llegue a manos de los clientes cumpliendo todas sus funcionalidades y quedando lo más libre posible de defectos. El proceso de pruebas cuenta con actividades bien definidas, comienza con la planificación de las pruebas, luego la ejecución, el control y por último la evaluación de las mismas. Todas estas actividades son llevadas a cabo por un proceso de retroalimentación, lo que permite que los errores encontrados se resuelvan de manera efectiva en iteraciones posteriores de forma tal que no afecten el tiempo de duración del proyecto y el costo del mismo.

Este capítulo tiene como objetivos principales abordar los temas fundamentales referidos a la ingeniería de pruebas, así como los conceptos y las definiciones expuestos por varios autores. Haciendo énfasis en algunos aspectos importantes relacionados con la calidad de software, tales como: los factores que miden la calidad de estos, así como las métricas asociadas; los niveles de calidad; los tipos de pruebas existentes, y el plan de pruebas; destacando algunos criterios generales, de los cuales se presentará su definición. Además se hará referencia a las definiciones que propone el Proceso Unificado de Desarrollo (RUP) o Rational Unified Process como metodología de desarrollo para este proceso.

1.2 Calidad de Software. Conceptos y definiciones

El término de la calidad de software es una preocupación a la que se dedican muchos esfuerzos. En la informática cuando se habla de este tema lo primero en que piensan las personas es en un producto que funcione bien, sin embargo hay que tener en cuenta que la calidad es mucho más que esto y para lograrla hay que realizar una serie de pasos durante todo el proceso de desarrollo del software, para que al finalizar se pueda decir que se cuenta con un producto de buena calidad. Sin embargo, cabe señalar que un software casi nunca es perfecto. Todo proyecto tiene como objetivo producir software de la mejor calidad posible, que cumpla, y si puede supere las expectativas de los usuarios.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Cuando se usa el término de calidad de software, muchas personas desconocen el verdadero significado que este encierra. Pudiéndose decir que es usado frecuentemente por muchas personas sin saber cuál es la magnitud de esta palabra. Para definir esta, primeramente se expone un concepto de que se entiende por calidad, de forma general.

Calidad: propiedad inherente a una cosa que permite compararla con la de su especie. Propiedad o conjunto de características de un elemento que le dotan de una ventaja competitiva. Es la totalidad de los rasgos y características de un producto o servicio que se sustenta en su habilidad para satisfacer las necesidades establecidas implícitas. **(1)**

En fin, se puede decir que esta palabra tiene muchas definiciones, pero la básica es la que establece que un producto o un servicio poseen calidad si satisface con las expectativas esperadas. El término calidad siempre será entendido de diferentes maneras por cada persona, ya que para unos la calidad residirá en un producto y para otros en el servicio posventa del mismo. Lo cierto es que nunca se llegará a definir exactamente lo que representa el término calidad a pesar de que últimamente este término se haya puesto de moda.

Teniendo en cuenta esta definición muchos autores han creado conceptos sobre la **Calidad de Software.** De ellos se escogieron los siguientes**:**

Conjunto de características de un producto o servicio que le confieren aptitud para satisfacer las necesidades del usuario o cliente. **(2)**

La capacidad de un conjunto de características inherentes de un producto, de los componentes del producto o de un proceso para satisfacer los requerimientos impuestos por los clientes. **(3)**

“El grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”. **(4)**

“Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario” .(5)

La calidad del software está dada por la calidad de los procesos usados para desarrollarlo y mantenerlo. (5)

Dentro del contexto de ingeniería de software, se tomará la definición de calidad en el software propuesta por la organización internacional de estándares (ISO/IEC DEC 9126), la cual la define como: la totalidad

de características de un producto de software que tienen como habilidad, satisfacer necesidades explícitas o implícitas.

Cuando se utiliza el término calidad en el proceso de desarrollo de software se hace alusión a los factores de un producto de este tipo que contribuyen a la satisfacción total de las necesidades de una persona específica o alguna organización. (1)

En cierta medida todos los conceptos expuestos hacen referencia a lo que es la **calidad de software** por lo que se puede definir de forma general que la calidad de software no es más que: el grado con el que un componente o un software cumplen con las necesidades del cliente, que cuenta con una serie de parámetros definidos y lo dotan de cualidades únicas, cumpliendo la totalidad de características de un producto de software, posibilitando satisfacer las necesidades explícitas o implícitas del usuario. (2)
Después de haber analizado el concepto de calidad, centrándose fundamental en el de calidad de software como tal, es necesario analizar otros conceptos muy importantes como es el caso de:

Las **pruebas de software**, en inglés *testing* o *beta testing,* que son los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. (2)
Otro aspecto que hay que tener en cuenta cuando se va analizar la calidad de un producto son las tareas que se van a llevar a cabo para garantizar esta, comúnmente llamado aseguramiento de la calidad. Antes de analizar que es el **aseguramiento de la calidad (QA)** por sus siglas de las palabras en inglés (Quality Assurance), es necesario primero considerar que la palabra "asegurar" implica afianzar algo, garantizar el cumplimiento de una obligación, transmitir confianza a alguien, afirmar, prometer, comprobar la certeza de algo, cerciorar; de acuerdo con esto, a través del aseguramiento, la organización intenta transmitir la confianza de un producto y/o servicio. (3)
La norma NMX-CC-001[2]:1995 define al aseguramiento de la calidad como el "conjunto de actividades planeadas y sistemáticas implantadas dentro del sistema de calidad, y demostradas según se requiera para proporcionar confianza adecuada de que un elemento cumplirá los requisitos para la calidad". (4)

Otra definición para esta expresión que es la que se escogerá para el sustento de esta investigación es la siguiente: “conjunto de acciones planificadas y sistemáticas que son necesarias para proporcionar la adecuada confianza de que un producto o servicio satisfará los requisitos dados sobre la calidad”. (5)

[2] *NMX-CC-001: 1995-IMNC (ISO 8402: 1994) Administración de la calidad y aseguramiento de la calidad - Vocabulario.*

Luego de haber realizado el análisis de estos conceptos vale la pena decir que cuando se va a comprobar la eficacia de un producto, es necesario tener en cuenta una serie de factores para poder justificar la calidad del mismo. Por esto se hace inevitable que estos sean examinados detalladamente con el fin de saber cuáles son y por qué precisamente con ellos se evalúa la calidad de un producto.

1.3 Factores que miden la calidad del software

Un aspecto muy importante del conocimiento a cerca de la calidad del software, son los factores por los cuales se puede medir la calidad de un producto. Estos se pueden dividir en tres grupos:

TABLA 1. FACTORES DE CALIDAD DE McCaLL

Aspecto que trata	Factor Calidad	Relacionado con …
Operación del Producto	Corrección	Grado en que un programa consigue los objetivos de la misión encomendada por el cliente.
	Fiabilidad	Probabilidad de operación libre de fallos de un programa de computadora en un entorno determinado durante un tiempo específico.
	Eficiencia	Cantidad de recursos de computadora y de código requeridos por un programa para llevar a cabo sus funciones.
	Integridad	Grado en que puede controlarse el acceso al software o a los datos, por personal no autorizado.
	Facilidad de uso	Esfuerzo requerido para aprender un programa, trabajar con él, preparar su entrada e interpretar su salida.
Revisión del producto	Facilidad de Mantenimiento	Esfuerzo requerido para localizar y arreglar un error en un programa.
	Flexibilidad	Esfuerzo requerido para modificar un programa operativo.
	Facilidad de prueba	Esfuerzo requerido para probar un programa de forma que se asegure que realiza la función requerida.

Transición del producto	Portabilidad	Esfuerzo requerido para transferir el programa desde un hardware y/o entorno de sistemas de software a otro
	Reusabilidad	Grado en que un programa (o partes de un programa) puede ser usado en otras aplicaciones.
	Facilidad de Interoperación	Esfuerzo requerido para acoplar un sistema a otro.

Tabla 1: Factores de calidad de MCCALL. (6)

Según (Pressman, 1995), basándose en los estudios de (McCall, 1977), los factores que afectan a la calidad del software se centran en tres aspectos importantes de un producto software: sus características operativas, su capacidad de soportar los cambios (revisión del producto) y su adaptabilidad a nuevos entornos (o transición del producto). En la Tabla 1 se muestran los factores de calidad definidos por McCall. Estos factores de calidad de McCall, a pesar de haberse definido hace mucho tiempo, conservan en gran medida su vigencia. Es difícil desarrollar medidas directas de los anteriores factores de calidad. Por tanto, se define un conjunto de métricas usadas para evaluar los factores de calidad. Según de qué factor se trate, se utilizarán unas determinadas métricas ponderadas para determinar ese factor. Las métricas que utiliza McCall para evaluar los distintos factores de calidad se muestran en la Tabla 2. (6)

Métricas propuestas por McCaLL	
Métrica	Relacionado con …
Facilidad de auditoría	La facilidad con que se puede comprobar la conformidad con los estándares.
Exactitud	La precisión de los cálculos y del control.
Normalización de las comunicaciones	El grado en que se usa el ancho de banda, los protocolos y las interfaces estándar.
Completitud	El grado en que se ha conseguido la total implantación de las funciones requeridas.
Concisión	Lo compacto que es el programa en términos de líneas de código.
Consistencia	El uso de un diseño uniforme y de técnicas de documentación a lo largo de todo el programa.
Estandarización en los datos	El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.
Tolerancia de errores	El daño que se produce cuando el programa detecta una situación errónea.

Eficiencia en la ejecución	El rendimiento en tiempo de ejecución de un programa.
Facilidad de expansión	El grado en que se puede ampliar el diseño arquitectónico, de datos o procedimental.
Generalidad	La amplitud de aplicación potencial de los componentes del programa.
Independencia del hardware	El grado en que el software es independiente del hardware sobre el que opera.
Instrumentación	El grado en que el programa muestra su propio funcionamiento e identifica errores que aparecen.
Modularidad	La independencia funcional de los componentes del programa.
Facilidad de operación	La facilidad de utilización de un programa.
Seguridad	La disponibilidad de mecanismos que controlen o protejan los programas o los datos.
Autodocumentación	El grado en que el código fuente proporciona documentación significativa.
Simplicidad	El grado en que un programa puede ser entendido sin dificultad.
Independencia del sistema de software	El grado en que el programa es independiente de características no estándar del lenguaje de programación, de las características del sistema operativo y de otras restricciones del entorno.
Facilidad de traza	La posibilidad de seguir la pista a la representación del diseño o de los componentes reales del programa hacia atrás, hacia los requisitos.
Formación	El grado en que el software ayuda a permitir que nuevos usuarios empleen el sistema.

Tabla 2: Métricas propuestas por MCCALL (6)

1.4 Aseguramiento de la calidad y pruebas de software

Cuando se menciona el término de calidad de software es necesario tratar dos conceptos muy importantes: como es el aseguramiento de la calidad, y pruebas de software (Software Testing), aunque las pruebas de software son una parte del proceso de aseguramiento de calidad. Las pruebas de software se integran dentro de las diferentes fases del ciclo de desarrollo del software dentro de la ingeniería de software. **(7)**

Para determinar el nivel de calidad de un producto se deben efectuar diferentes tipos de pruebas de manera tal que se permita comprobar el grado de cumplimiento que tiene el sistema, respecto a las cualidades funcionales que se definieron en las etapas iniciales del proceso. Con las pruebas de software o testing se identifican los posibles fallos de implementación, calidad, o usabilidad de un programa. Sin

embargo: "El testing puede probar la presencia de errores pero no la ausencia de ellos[3]". Esta básicamente es una etapa en el desarrollo de software consistente en probar las aplicaciones construidas a través de todo el proceso de desarrollo. **(7)**

Las pruebas de software son una parte importante dentro de casi todos los modelos conocidos de ciclo de vida del software y por otro lado, QA se refiere a asegurar la calidad en cada una de las fases de la elaboración de un producto final, cualquiera que este sea. En el caso de QA de software, se referirá a asegurar la calidad de los resultados de cada una de las fases del ciclo de vida del software y con esto, asegurar la calidad del producto final. Para cumplir con este aseguramiento se deberán definir estándares y establecer procedimientos contra los cuales se pueda comparar lo alcanzado durante cada una de las fases. La idea es que mientras más temprano se detecten las fallas, menor será el costo (monetario, de tiempo, recursos y calidad) de repararlas y mayor la calidad del producto final. **(8)**

Una vez conocidas estas definiciones es muy factible analizar algunas de las semejanzas y diferencias relacionadas con las pruebas de software y QA.

Diferencias entre pruebas de software y QA	
Pruebas de software	Aseguramiento de la calidad (QA)
Utilizan casos de pruebas para ser ejecutados	Utilizan los estándares y procedimientos establecidos para cada una de las fases del ciclo de vida del software.
Permite verificar y afirmar la calidad del producto final, el software	Permite verificar y afirmar la calidad del producto final, el software
Define un conjunto de actividades a realizar dentro del ciclo de vida del software para mejorar y asegurar la calidad del mismo.	Define un conjunto de actividades a realizar dentro del ciclo de vida del software para mejorar y asegurar la calidad del mismo.

Tabla 3: diferencias entre pruebas de software y QA (6)

Tanto las pruebas de software como el aseguramiento de la calidad necesitan de una guía para que se lleven a cabo con la máxima eficiencia posible, aquí es donde juega un papel de suma importancia la metodología de desarrollo que se utilizará en el proyecto. Esta es la encargada de administrar de cierta manera el estado de las pruebas de software y el QA, definiendo el cuándo y el tiempo que se le dedicará a cada una de las pruebas.

1.5 Metodología de desarrollo de software

[3] *E. W. Dijkstra (11 de mayo de 1930 - 6 de agosto de 2002). Fue físico, matemático y uno de los padres de la programación.*

Las metodologías de desarrollo de software es uno de los términos que más confusión produce tanto en estudiantes como en profesionales involucrados en estos procesos, definen un conjunto de criterios que guían la forma en que se aplica la ingeniería del software en un proyecto productivo; en otras palabras, es la base para la construcción de un proyecto de software, la etapa fundamental para lograr los objetivos buscados con dicho proyecto. No es que haya una metodología superior a las demás, todas las metodologías son, en esencia, buenas. Obviamente, las más modernas responden a problemas y necesidades actuales. La ausencia de metodología en el desarrollo de un proyecto de software garantiza con seguridad también la poca o nula calidad del mismo.

La obtención de un software con calidad implica la utilización de metodologías o estándares para el análisis, diseño, programación y prueba de este, que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, que a la vez eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software. Su objetivo es controlar de manera transparente todo el proceso de desarrollo, fundamentalmente permite producir sistemas en el tiempo planificado para este y con el costo estimado, esto es básico pues muchas veces se planea hacer algo que finalmente toma más tiempo de lo planeado. [6]

Actualmente con el creciente desarrollo tecnológico y con el surgimiento de nuevos modelos de producción, han ido apareciendo nuevas metodologías de proceso de desarrollo, estos procesos han ido tomando características marcadas, lo que ha conllevado a agruparlos en dos grupos, los llamados "métodos robustos" y los "métodos ágiles", la diferencia más notable entre estos es que mientras los métodos robustos intentan obtener los resultados ayudándose principalmente de la documentación ordenada, los métodos agiles tienen como base de sus resultados a la comunicación e interacción directa con todos los usuarios involucrados en el proceso.

Según estudios realizados con anterioridad se decide que la metodología de desarrollo que se utilizará durante todo el proceso de desarrollo de construcción del proyecto Captura y Catalogación de Medias (CCM) es la mitología RUP.

¿Por qué aplicar RUP en el proyecto CCM?

Se selecciona RUP porque sus características son factibles para la realización del sistema, primero porque el Proceso Unificado es una propuesta de proceso para el desarrollo de software orientado a objetos que utiliza *lenguaje modelado unificado (UML)* como único lenguaje para describir el proceso. Está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas. Además el sistema se va desarrollando y documentando al mismo tiempo por si algún miembro del equipo no puede seguir con el trabajo el que ocupe su lugar tenga por donde guiarse y conozca que fue lo que se hizo. **(9)**

Ahora se hace factible analizar qué es lo que plantea esta metodología para su aplicación y sus principales elementos.

1.5.1 Proceso Unificado de Desarrollo (RUP)

Dentro de las metodologías fuertes la que más se destaca es el Proceso Unificado de Desarrollo (RUP). Esta es una metodología para la ingeniería de software que va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. El resultado es un proceso basado en componentes, dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. [4]

La metodología RUP[4] es utilizada en la construcción de proyectos nuevos, pero también se puede aplicar para actualizaciones de proyecto ya realizados, y recomiendan adoptarla de forma gradual. RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML[5] que no es más que el lenguaje para visualizar, especificar, construir y documentar los artefactos que se generan durante todo el ciclo de RUP. [4]

Como RUP es un proceso, en su modelación definen como sus principales elementos

Elementos	Descripción
Trabajadores	Define el comportamiento y responsabilidades de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
Actividades	Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
Artefactos	Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
Flujo de actividades	Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

Tabla 4: Principales elementos que define RUP (8)

[4] *Rational Unified Process*

[5] *Unified Modeling Language Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo.*

RUP está enfocado a cualquier tipo de proyecto, se basa en la documentación generada en cada uno de sus cuatro fases: Inicio (puesta en marchar), 2. Elaboración (definición, análisis y diseño), 3. Construcción (implementación) y 4. Transición (fin del proyecto y puesta en producción) en las cuales se ejecutarán varias iteraciones (según el tamaño del proyecto) y complejidad del mismo. A continuación se muestra la imagen de los flujos de trabajo que define RUP para cada unas de las fases. **(8)**

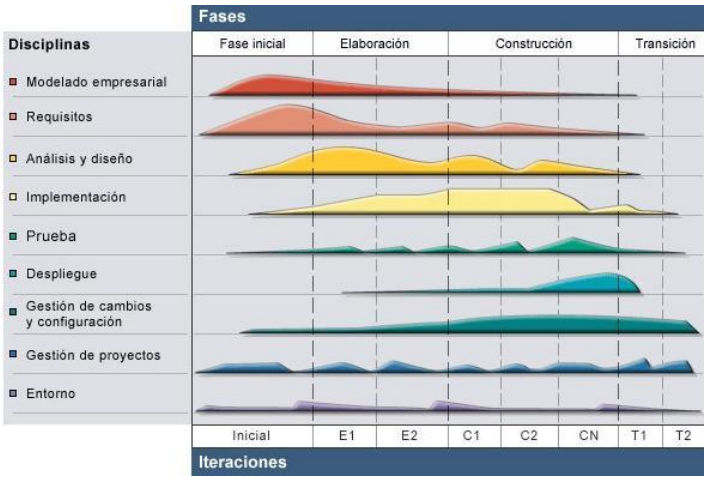


Figura 1: Diagrama flujo de trabajos RUP (8)

RUP se basa en casos de uso (Use Case) para describir lo que se tiene y lo que se espera del software, está muy orientado a la arquitectura del sistema a implementar, documentándose de la mejor manera, basándose en UML.

Para poder usar RUP antes hay que adaptarlo a las características de la empresa, y medir de manera exacta el tiempo, costos y todos los demás recursos involucrados en el proceso. (8)

RUP ofrece un escenario formidable para la realización del diseño de los casos de prueba ya que por ser una metodología guiada por casos de usos, en ellos se recoge detalladamente toda la información de las funcionalidades del sistema en general. Esto garantiza que a la hora de realizar los diseños de casos de prueba no se quede ninguna funcionalidad sin probar, esta ventaja no lo garantiza ninguna otra metodología, ya que carecen de documentación en gran escala.

RUP como metodología de desarrollo al transitar por cada fase, ya ante descrita, define una serie de disciplinas dentro de las cuales se encuentran las pruebas. Disciplina que está presente en cada una de las fases, alcanzando mayor peso en la fase de construcción. Esta disciplina tiene una importancia muy alta, ya que es la que va a garantizar la calidad del producto fin, por esto es que es necesario analizar detalladamente cómo es que funciona este proceso. (8)

1.5.2 Proceso de pruebas de RUP.

Dentro de las metodologías pesadas RUP es la más popular y usada que existe, esta describe cómo planear y ejecutar las pruebas teniendo en cuenta un grupo de artefactos, actividades y trabajadores.
Los trabajadores definidos en RUP para la fase de prueba son:

Diseñador de pruebas: Planifica las pruebas. Es responsable de la integridad del modelo de pruebas asegurando que el modelo cumple con su propósito. Del modelo de pruebas deben definir y describir los casos de prueba y los procedimientos de prueba. Además son los responsables de la evaluación de las pruebas de integración y de sistema cuando éstas se ejecuten. (8)

Ingeniero de componentes: Son responsables de los componentes de pruebas que automatizan algunos de los procedimientos de pruebas. (8)

Ingeniero de pruebas de integración: Son los responsables de realizar las pruebas de integración, las cuales se realizan para verificar que los componentes integrados en una construcción funcionan correctamente juntos, y se derivan a menudo de los casos de pruebas que especifican cómo probar realizaciones de caso de uso de diseño. Debe documentar los defectos en las pruebas de integración. (8)

Ingeniero de pruebas de sistema: Son los responsables de realizar las pruebas de sistema sobre los ejecutables. Estas pruebas se llevan a cabo fundamentalmente para verificar las interacciones entre los actores y el sistema. Estas pruebas se derivan a menudo de los casos de pruebas que especifican cómo probar los casos de uso, también se usan para probar el sistema como un todo. Debe documentar los defectos en las pruebas de sistema. (8)

Dentro del plan de pruebas RUP define una serie de actividades, mediante las cuales se va a guiar todo el proceso de pruebas. Estas tienen una gran importancia pues en ellas recae todo el peso del flujo de pruebas y si no se realizan cada una de ellas no se podrá obtener una buena evaluación de pruebas. A continuación se muestran cuales son estas actividades.

1.5.3 Actividades fundamentales del flujo de trabajo de prueba

Las actividades fundamentales que se desarrollan en el flujo de trabajo de pruebas son las siguientes. (6)

1. Planificar las pruebas.
2. Diseñar las pruebas: Dentro de esta actividad se realizan pruebas de integración y de sistema.
3. Implementar prueba.
4. Realizar pruebas de integración.

5 Realizar pruebas de sistema.

6 Evaluar pruebas.

Una vez realizadas estas actividades traerá consigo que se generen un grupo de artefactos[6], que tiene gran importancia para el desarrollo de las pruebas. A continuación se mencionan dichos artefactos.

1.5.4 Artefactos del flujo de trabajo de prueba.

El Proceso Unificado de Desarrollo de software plantea 11 artefactos para el flujo de trabajo de prueba los cuales deben ser generados dentro de esta fase. Los cuales se muestran a continuación. **(8)**

1 Plan de prueba.

2 Estrategia de prueba.

3 Arquitectura de automatización de la prueba.

4 Configuración del entorno de la prueba.

5 Script de prueba.

6 Casos de Prueba.

7 Suite de Pruebas.

8 Especificación de interfaz de la prueba.

9 Resumen de evaluación de las pruebas.

10 Resultado de la prueba.

11 Datos de Pruebas.

1.5.4.1 Documentos del proceso de pruebas utilizados en la UCI

En el proceso de prueba a realizar al producto CCM quedan definidos los mismos entregables que define el Centro para la Excelencia en el Desarrollo de Proyectos Tecnológicos de la UCI (Calisoft). **(10)**

1. El plan de prueba.
2. Casos de prueba.
3. El documento de No Conformidades. (Resultado de la prueba)

Todos los otros artefactos que se mencionan a pesar de que no tengan un entregable tangible se desarrollarán en este trabajo como son: la estrategia de pruebas, configuración del entorno de pruebas, script de pruebas, suite de pruebas, configuración del entorno de la prueba, resumen de evaluación de las pruebas y datos de pruebas. No siendo así con arquitectura de automatización de la prueba y Especificación de interfaz. El primero no se realiza pues como no se va utilizar ninguna herramienta para

[6] *modelos, reportes, documentos*

aplicar pruebas al producto CCM y el segundo porque este solo se utiliza en situaciones en que los aspectos del sistema que normalmente no tienen visibilidad deben observarse, o donde el control del software es necesario de un modo normalmente no disponible a través de la interfaz estándar.

Dentro de estos artefactos el plan de pruebas tiene como propósito general explicitar el alcance del proyecto. Este se realiza con el objetivo principal de establecer la cronología y condiciones para la aplicación de las pruebas, con el fin de obtener un producto que tenga una buena aceptación por parte del cliente. Por lo que se convierte en una necesidad analizar detalladamente cuales son los pasos, características y elementos que hay que tener en cuenta a la hora de realizar un plan de pruebas.

1.5.4.2 Plan de pruebas

Cuando se va a realizar un proyecto sea cual sea su magnitud, la calidad es un factor imprescindible en el mismo, para lograr este objetivo es importante la realización de un plan de pruebas, con el que se describe la estrategia que se utilizará, los recursos que serán empleados y planificación de estas. Esta estrategia incluye la definición del tipo de pruebas a realizar para cada iteración y sus objetivos, el nivel de cobertura que tendrá la misma y el porcentaje de las pruebas que deberían ejecutarse con un resultado específico. Cada una de ellas debe dejar claro qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad). Además, un plan de prueba debe especificar en qué consiste la misma (hasta el último detalle de cómo se ejecuta), y debe definir cuál es el resultado que se espera este tiene como objetivo principal el de explicitar el alcance del proyecto, recursos requeridos, responsable de cada actividad y tener un manejo de los principales riesgos que pueden atentar contra el producto final. (7)

Un plan de pruebas incluye los siguientes elementos:

1. **Identificador del plan:** como todo artefacto del desarrollo, está sujeto a control de configuración, por lo que debe distinguirse adicionalmente la versión y fecha del plan. (8)
2. **Alcance**: indica el tipo de prueba y las propiedades/elementos del software a ser probado.
3. **Ítems a probar:** indica la configuración a probar y las condiciones mínimas que debe cumplir para comenzar a aplicarle el plan. Por un lado, es difícil y riesgoso probar una configuración que aún reporta fallas; por otro lado, si se espera a que todos los módulos estén perfectos, pueden detectarse fallas graves demasiado tarde. (8)
4. **Estrategia:** describe la técnica, patrón y/o herramientas a utilizarse en el diseño de los casos de prueba. Por ejemplo, en el caso de pruebas integración de un procedimiento, esta sección podría indicar: "Se aplicará la estrategia caja-negra de fronteras de la precondición". En lo posible la estrategia debe precisar el número mínimo de casos de prueba a diseñar, por ejemplo, 100% de las

fronteras, 60% de los caminos ciclomáticos. La estrategia también explicita el grado de automatización que se exigirá, tanto para la generación de casos de prueba como para su ejecución. (8)

5. **Categorización de la configuración:** explicita las condiciones bajo las cuales, el plan debe ser:
   - ✓ Suspendido
   - ✓ Repetido
   - ✓ Culminado

   En algunas circunstancias el proceso de prueba debe suspenderse en vista de los defectos o fallas que se han detectado. Al corregirse los defectos, el proceso de prueba previsto por el plan puede continuar, pero debe explicitarse a partir de qué punto, ya que puede ser necesario repetir algunas pruebas. Los criterios de culminación pueden ser tan simples como aprobar el número mínimo de casos de prueba diseñados o tan complejos como tomar en cuenta no sólo el número mínimo, sino también el tiempo previsto para las pruebas y la tasa de detección de fallas. (8)
6. **Tangibles:** explicita los documentos a entregarse al culminar el proceso previsto por el plan. (8)
7. **Recursos:** especifica las propiedades necesarias y deseables del ambiente de prueba, incluyendo las características del hardware, el software de sistemas, cualquier otro software necesario para llevar a cabo las pruebas, así como la colocación específica del software a probar y la configuración del software de apoyo. (8)
8. **Calendario:** esta sección describe los hitos del proceso de prueba y el grafo de dependencia en el tiempo de las tareas a realizar. (8)
9. **Manejo de riesgos:** explicita los riesgos del plan, las acciones mitigantes y de contingencia. (8)
10. **Responsables:** especifica quién es el responsable de cada una de las tareas previstas en el plan. (8)

Como se ha plasmado con anterioridad el plan de pruebas es el que va a guiar todo el proceso de prueba durante el desarrollo de software. Pero al igual que otros procesos tiene que estar guiado por diferentes estándares o modelos de calidad ya existentes a escala internacional. Por lo que se convierte en necesidad analizar cuáles son estos modelos.

1.6 Elementos fundamentales de proceso de prueba.

1.6.1 Niveles de prueba

Prueba de desarrollador: La prueba de desarrollador indica los aspectos de diseño e implementación de las pruebas más adecuadas que debe llevar a cabo el equipo de desarrolladores, a diferencia de la prueba independiente. En la mayoría de los casos, la ejecución de la prueba se produce inicialmente con el grupo de pruebas de desarrollador que la diseñó e implementó; aunque es recomendable que los

desarrolladores creen las pruebas de forma que estén disponibles para que las ejecuten grupos de pruebas independientes. **(8)**

Prueba independiente: Las pruebas independientes indican el diseño y la implementación de la prueba realizada más adecuadamente por alguien ajeno al equipo de desarrolladores. Puede considerar esta distinción un súper conjunto, que incluye validación y verificación independientes. En la mayoría de los casos, la ejecución de la prueba se produce inicialmente con el grupo de pruebas independientes que la diseñó e implementó; aunque los verificadores independientes deberían crear sus pruebas de forma que estén disponibles para que las ejecuten los grupos de pruebas de desarrollador. **(8)**

Prueba de unidad: La prueba de unidad se centra en la verificación de los elementos más pequeños del software que se puedan probar. Normalmente, las pruebas de unidad se aplican a componentes representados en el modelo de implementación para verificar que se cubren los flujos de control y los flujos de datos y que funcionan como se esperaba. El implementador realiza la prueba de unidad mientras se desarrolla la unidad. Los detalles de la prueba de unidad se describen en la disciplina de implementación. **(8)**

Prueba de integración: Las pruebas de integración se realizan para garantizar que los componentes del modelo de implementación funcionan correctamente cuando se combinan para ejecutar un guión de uso. El destino de la prueba es un paquete o un conjunto de paquetes del modelo de implementación. A menudo, los paquetes que se combinan proceden de diferentes empresas de desarrollo. Las pruebas de integración exponen el estado incompleto o los errores de las especificaciones de la interfaz del paquete. **(8)**

En algunos casos, los desarrolladores presuponen que otros grupos, como los verificadores independientes, son los encargados de realizar las pruebas de integración. Esta situación supone un riesgo para el proyecto de software y, en última instancia, para la calidad del software, puesto que:

- Las áreas de integración son un punto común de fallo del software.
- Las pruebas de integración realizadas por verificadores independientes suelen utilizar técnicas de caja negra y tratar componentes de software más grandes.

Prueba del sistema: Normalmente, la prueba del sistema se realiza cuando el software funciona en su totalidad. Un ciclo vital repetitivo permite que las pruebas del sistema se realicen mucho antes, en cuanto se hayan implementado subconjuntos bien formados del comportamiento de guiones de uso. Normalmente, el destino son los elementos en funcionamiento de extremo a extremo del sistema. **(8)**

Prueba de aceptación: La prueba de aceptación del usuario es la última acción de prueba antes de desplegar el software. El objetivo de la prueba de aceptación es comprobar si el software está preparado y lo pueden utilizar los usuarios para realizar las funciones y tareas para las que se diseñó. **(8)**

Luego de ver analizado todos los niveles de prueba por lo que puede pasar un software destacando los principales objetivos que tienen en cada uno de estos, es necesario estudiar cuales son los tipos de pruebas que se aplican en cada uno de estos niveles. En el siguiente epígrafe se mostrarán dichas pruebas, enfocado esencialmente a destacar cual es el objetivo fundamental de cada prueba.

1.6.2 Tipos de prueba

Las pruebas de software informático implican mucho más que la simple evaluación de las funciones, la interfaz y las características de tiempo de respuesta de un destino de la prueba. Las pruebas adicionales deben centrarse en características y atributos, como el destino de la prueba.

Para conseguir esto, deben implementarse y ejecutarse muchos tipos diferentes de pruebas, la idea es que cuantas más pruebas se le apliquen a un software, mayor será el grado de calidad que tendrá el mismo. Cada tipo de prueba tiene un objetivo específico y cada técnica se centra en la prueba de uno o varios atributos o características del destino de la prueba. A continuación de muestra una tabla donde se recogen los diferentes tipos de prueba existentes: (8)

Tipos de Pruebas que se aplican al software (8)	
Dimensión de calidad	**Tipo de pruebas**
Funcionalidad	**Prueba de función:** pruebas que se centran en la validación de funciones del destino de la prueba, proporcionan los guiones de uso, los métodos y los servicios necesarios. Esta prueba se implementa y se ejecuta en diferentes destinos de la prueba, incluidas las unidades, las unidades integradas, las aplicaciones y los sistemas.
	Prueba de seguridad: pruebas que se centran en garantizar que los datos del destino de la prueba (o sistemas) sólo son accesibles para los actores a los que se dirigen. Esta prueba se implementa y ejecuta en varios destinos de la prueba.
	Prueba de volumen: pruebas que se centran en la verificación de la capacidad del destino de la prueba para manejar grandes cantidades de datos, ya sean de entrada y salida o residentes, en la base de datos. La prueba de volumen incluye estrategias de prueba como la creación de consultas que devolverán el contenido completo de la base de datos, o que tendrán tantas restricciones que no devolverán ningún dato, o en las que la entrada de datos tiene la cantidad máxima de datos para cada campo.

Fiabilidad	**Prueba de integridad:** pruebas que se centran en la evaluación de la fuerza del destino de la prueba (resistencia a los errores) y la conformidad técnica del lenguaje, la sintaxis y la utilización de recursos. Esta prueba se implementa y se ejecuta en diferentes destinos de la prueba, incluidas las unidades y las unidades integradas.
	Prueba de estructura: pruebas que se centran en la evaluación de la adherencia del destino de la prueba a su diseño y formación. Normalmente, esta prueba se realiza en aplicaciones habilitadas para web y garantiza que todos los enlaces están conectados, se muestra el contenido adecuado y no hay ningún contenido huérfano.
	Prueba de tensión: se trata de un tipo de prueba de fiabilidad que se centra en la evaluación de cómo responde el sistema en circunstancias anormales. Las tensiones del sistema pueden ser cargas de trabajo extremas, memoria insuficiente, servicios y hardware no disponible o recursos compartidos limitados. Estas pruebas suelen realizarse para saber mejor cómo y en qué áreas fallará el sistema, de forma que se puedan planificar y presupuestar los planes de contingencia y el mantenimiento de las actualizaciones con bastante antelación.
	Prueba de puntos de referencia: se trata de un tipo de prueba de rendimiento que compara el rendimiento de un destino de la prueba nuevo o desconocido con una referencia conocida, carga de trabajo y sistema.
	Prueba de contienda: pruebas que se centran en la validación de la capacidad del destino de la prueba para manejar de forma aceptable varias demandas del actor en el mismo recurso (registros de datos, memoria, etc.).
Rendimiento	**Prueba de carga:** se trata de un tipo de prueba de rendimiento que se utiliza para validar y evaluar la aceptabilidad de los límites operativos de un sistema bajo cargas de trabajo variables, mientras el sistema que se está probando permanece igual. En algunas variantes, la carga de trabajo permanece igual y se modifica la configuración del sistema que se está probando. Las medidas suelen tomarse en función del rendimiento de la carga de trabajo y el tiempo de respuesta de las transacciones en línea. Las variaciones de la carga de trabajo suelen incluir la emulación del pico y el promedio de cargas de trabajo que se producen dentro de la tolerancia operativa normal.
	Perfil de rendimiento: se trata de una prueba en la que se controla el perfil de tiempo del destino de la prueba, incluidos el flujo de la ejecución, el acceso de datos, las llamadas del sistema y de funciones para identificar y tratar los cuellos

	de botella de rendimiento y los procesos ineficaces.
	Prueba de configuración: pruebas que se centran en garantizar que las funciones del destino de la prueba son las adecuadas en diferentes configuraciones de hardware y software. Esta prueba también se puede implementar como una prueba de rendimiento del sistema.
Capacidad de soporte	**Prueba de instalación:** pruebas que se centran en garantizar que el destino de la prueba se instala correctamente en diferentes configuraciones de hardware y software, y en condiciones diferentes (como, por ejemplo, espacio de disco insuficiente o interrupciones de la alimentación). Esta prueba se implementa y ejecuta en aplicaciones y sistemas.

Tabla 5 : Tipos de Pruebas que se aplican al software

Tanto los niveles de prueba como los tipos de prueba que se mencionan con anterioridad para ser llevados a cabo de una forma eficiente, necesitan de procedimientos o técnicas que lo guíen en su ejecución, por lo que será una necesidad analizar cada una de estas técnicas.

1.7 Técnicas de diseño de pruebas

Las estrategias de pruebas del software se integran a las técnicas del diseño de casos de prueba, en una serie de pasos planificados que dan como resultado una correcta construcción del software, con tal grado de confianza que se detectarán la mayor parte de los errores existentes en él (nunca se puede decir que el producto está libre de errores).

Un aspecto primordial de esta parte es la definición de los casos de prueba. Pressman define los casos de pruebas como “un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo en particular. **(2)**

Para este diseño de casos de prueba se identifican 3 enfoques de pruebas:

- El enfoque estructural o de caja blanca consiste en centrarse en la estructura interna (implementación) del programa para elegir los casos de prueba. Este enfoque se muestra en la ***figura 2***.
- El enfoque funcional o de caja negra consiste en estudiar la especificación de las funciones, la entrada y la salida para derivar los casos. Este enfoque se muestra en la ***figura 4***.
- El enfoque aleatorio, consistente en utilizar modelos (en muchas ocasiones estadísticos) que representen las posibles entradas al programa para crear a partir de ellos los casos de prueba.

A continuación se describe con detalle los métodos de caja blanca y caja negra:

1.7.1 Técnica de prueba de caja blanca

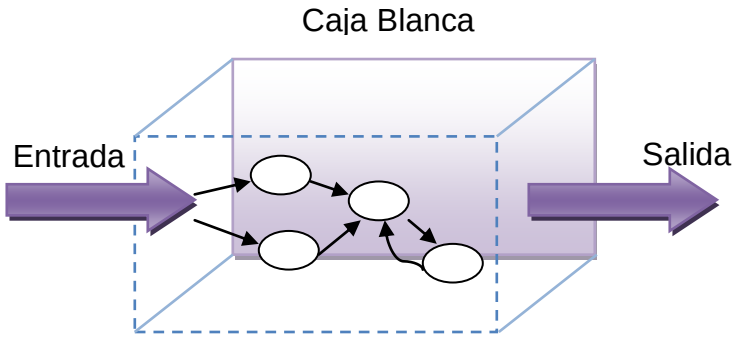


Figura 2: Prueba de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo, ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los ciclos en sus límites y con sus límites operacionales, y ejerciten las estructuras internas de datos para asegurar su validez. (11)

Con este método se determina cuáles son los casos de prueba a partir del código fuente del software y se utilizan las especificaciones para determinar el resultado esperado del caso. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa.

La prueba de caja blanca del software se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y\o ciclos. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado. **(11)**

Mediante este método el ingeniero de software puede obtener casos de prueba que:

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

Algunas técnicas empleadas en las pruebas de caja blanca son los siguientes:

1. **Prueba del camino básico:** Es una técnica que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizarán que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. **(7)**

Este tipo de técnica utiliza para su ejecución la construcción de un: **Grafo de flujo o grafo del programa:** este representa el flujo de control lógico de un programa y se utiliza para trazar más fácilmente los caminos de éste. (Cada nodo representa una o más sentencias procedimentales y cada arista representa el flujo de control). **(7)**

Para construir el grafo se debe tener en cuenta la notación para cada una de las instrucciones correspondientes como se muestra a en la figura a continuación.

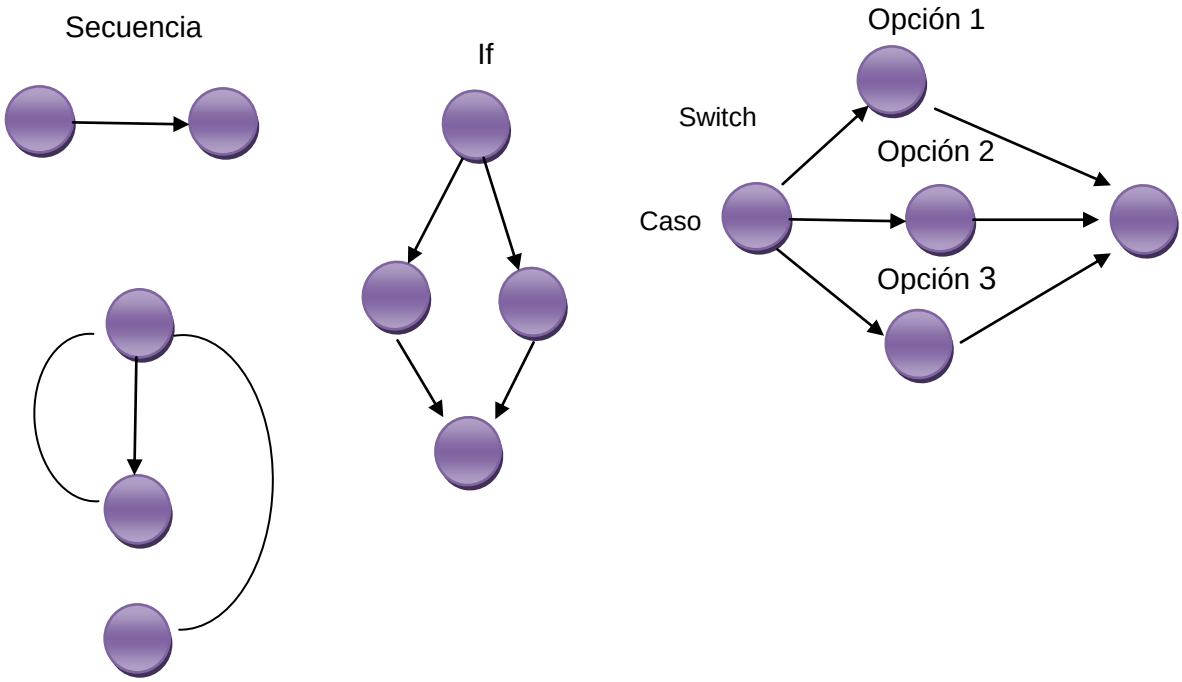


Figura 3: Notación de grafos de flujo para las instrucciones: Secuenciales, If, While, Switch

Para contar el número de ciclos diferentes que se siguen en un fragmento de código de un programa habiendo creado una rama imaginaria desde el nodo de salida al nodo de entrada se utiliza la Complejidad Ciclomática (Cyclomatic Complexity), esta es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Es una de las métricas de software más ampliamente aceptada, ya que ha sido concebida para ser independiente del lenguaje. **(7)**

Esta métrica, propuesta por Thomas McCabe[7], se basa en el diagrama de flujo determinado por las estructuras de control de un determinado código. De dicho análisis se puede obtener una medida cuantitativa de la dificultad de crear pruebas automáticas del código y también es una medición orientativa de la fiabilidad del mismo.

El resultado obtenido en el cálculo de la complejidad Ciclomática define el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. **(7)**

La medida resultante puede ser utilizada en el desarrollo, mantenimiento y reingeniería para estimar el riesgo, costo y estabilidad. (12)

Se calcula de la siguiente manera:

1ra vía V (G)= Número de Regiones.

2da vía V (G)= Número de Aristas – Número de Nodos + 2

3ra vía V (G)= Número de Nodos Predicados +1

Camino independiente: cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición. (12)

2. **Prueba de la estructura de control:** Dentro de éste tipo de prueba se contempla el método del camino básico mencionado anteriormente pero además existen otras pruebas asociadas que permiten ampliar la cobertura de la prueba y mejorar su calidad. Estas son: (12)

- **Prueba de condición:** es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa. Algunos conceptos empleados alrededor de esta prueba son los siguientes:
- **Condición simple:** es una variable lógica o una expresión relacional (E1 < operador - relacional > E2).
- **Condición compuesta:** está formada por dos o más condiciones simples, operadores lógicos y paréntesis. En general los tipos de errores que se buscan en una prueba de condición, son los siguientes:
- **Error en operador lógico** (existencia de operadores lógicos incorrectos, desaparecidos, sobrantes), error en variable lógica, error en paréntesis lógico, error en operador relacional, error en expresión aritmética.

3. **Prueba del flujo de datos:** selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. (12)

[7] *Propone la métrica de la complejidad Ciclomática en 1976*

4. **Prueba de bucles:** es una técnica que se centra exclusivamente en la validez de las construcciones de bucles (bucles simples, anidados, concatenados y no estructurados). (12)

A continuación se muestra en qué consiste el método de caja negra.

1.7.2 Técnica de prueba de caja negra.

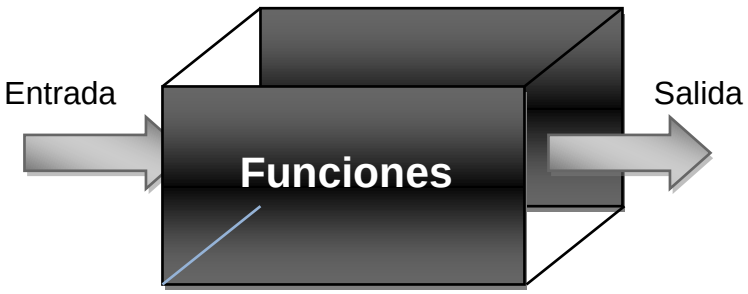


Figura 4: Prueba de caja negra

Las pruebas de caja negra se centran fundamentalmente en los requisitos funcionales del software. Es decir, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Se trata de un enfoque que intenta descubrir diferentes tipos de errores que no se encuentran con los métodos de caja blanca. Este método de prueba es realizado a nivel de sistema es decir no se tiene ninguna relación con el código del producto.

La prueba de caja negra, intentan encontrar errores de las siguientes categorías:

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

Métodos de prueba basados en grafos: en este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. **(13)**

Partición equivalente: Método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de

forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. **(13)**

Análisis de valores límite: Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límite. **(13)**

Prueba de la tabla ortogonal: hay aplicaciones donde el número de parámetros de entrada es pequeño y los valores de cada uno de los parámetros están claramente delimitados. Cuando estos números son muy pequeños (por ejemplo, 3 parámetros de entrada tomando 3 valores diferentes), es posible considerar cada permutación de entrada y comprobar exhaustivamente el proceso del dominio de entrada. En cualquier caso, cuando el número de valores de entrada crece y el número de valores diferentes para cada elemento de dato se incrementa, la prueba exhaustiva se hace impracticable. **(13)**

Adivinando el error: dado un programa particular, se conjetura, por la intuición y la experiencia, ciertos tipos probables de errores y entonces se escriben casos de prueba para exponer esos errores. Es difícil dar un procedimiento para esta técnica puesto que es en gran parte un proceso. **(13)**

Los métodos de prueba del software tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con el menor tiempo y esfuerzo posible. Pero por muy eficientes que sean, son métodos manuales, que en cierta medida no nos garantiza un 100% de veracidad de la calidad de un producto. Desde etapas iníciales de un producto informático siempre se tiene en cuenta garantizar la calidad del mismo a máxima escala. Este punto no siempre se ha tomado con tanta seriedad como se toma hoy día. Por esto es que se hace necesario analizar cuál es el estado actual de las pruebas de software.

1.8 Conclusiones

El análisis realizado en este capítulo ha demostrado de manera clara, la importancia que tiene calidad del software, a lo que contribuyó el abarcamiento de los temas tratados. Se analizó todo lo referente al proceso de pruebas en un software y los factores por los que se ve afectada la calidad de un producto, garantizando con esto tener una mejor visión ante los posibles fallos que puede tener un software.

Se estudiaron los niveles de prueba, tipos de pruebas, así como de los métodos y técnicas que se emplean para realizar las pruebas a cada uno de estos niveles, teniendo con esto la herramienta fundamental para la aplicación de las pruebas pertinentes al producto CCM.

Se enfatizó en la metodología de desarrollo que el proyecto seleccionó con anterioridad (RUP), dando a conocer cuáles eran las ventajas que esta ofrecía para el desarrollo de las pruebas al producto. También se mostró cuales son los principales artefactos, trabajadores y actividades que están estrechamente relacionados con el flujo de pruebas. Analizando también los modelos de calidad por los cuales se guiará todo el proceso de pruebas y la importancia que tiene la aplicación de los mismos.

CAPÍTULO 2
Diseño y Aplicación de Pruebas

2.1 Introducción

En el capítulo anterior se realizó un análisis exhaustivo relacionado con los principales conceptos relacionados a las pruebas de software. Todo lo anteriormente planteado servirá de guía para el diseño y aplicación de pruebas al producto Captura y Catalogación de de Medias (CCM). Es aquí donde se estudiará la distribución del proyecto CCM, como su estructura, subsistemas, funcionalidades. En el mismo se precisará un plan de pruebas como base para todo el proceso que se llevará a cabo, así como la estrategia de pruebas a seguir y la configuración del entorno de pruebas, especificando cada uno de los elementos de las pruebas definidos. Es aquí donde quedará plasmado el diseño de caso de pruebas de todos casos de usos que serán sometidos a prueba.

2.2 Sistema de Captura y Catalogación de Medias (CCM)

El Sistema de Captura y Catalogación de Medias (CCM) es un producto que se está desarrollando en el centro de producción de la facultad 9. Este se identifica fundamentalmente por la captura y catalogación de medias. Y para que este producto tenga la calidad que el cliente necesita debe cumplir con los requisitos que se mencionan a continuación.

Este producto permite: **(12)**

- Captura de Señales de Video según planificaciones definidas.
- Permite un fácil cambio de la señal que se está capturando en cada momento
- Permite almacenar los ficheros de video capturados en un archivo, por lo que se pueden volver a transmitir, si se está interesado.
- Permite la grabación de señales de audio o radio FM.
- Permite la visualización de los materiales audiovisuales almacenados.
- Permite la catalogación de los materiales audiovisuales almacenados.
- Permite transcribir ficheros de audio.
- Permite almacenar los recursos audiovisuales: video y audio.
- Permite crear reportes sobre la actividad de los usuarios en el sistema y el estado del servidor de medias.

Este sistema está estructurado en cuatro subsistemas: **(13)**

- Subsistema de Captura y Transcripción de Audio
- Subsistema de Captura de Video
- Subsistema de Catalogación
- Subsistema de Administración y Configuración.

2.3 Características a evaluar

Las características a evaluar en un software se escogen teniendo en cuenta el tipo de producto que sea y lo que se pretenda probar con la prueba a realizar. Hay algunas características que siempre se deben tener en cuenta, como por ejemplo, si la aplicación cumple con las funcionalidades requeridas desde el comienzo, si no tiene errores a la hora de ejecutarlo, si es comprensible para el usuario, si se puede usar correctamente con facilidad y si cuenta con una documentación detallada y que corresponda con la aplicación. **(14)**

También se pueden medir aspectos generales que tienen que ver con el código del programa y permiten conocer si dicho código está bien estructurado y es reutilizable. Por ello es importante comprobar que la aplicación sea fácil de mantener para garantizar que se mantenga activa por mucho tiempo. **(14)**

Para tener un control exacto de algún proceso, este debe ser debidamente concebido, por esto es sumamente importante realizar el plan de pruebas ya que va a ser la principal guía que va a tener el grupo de calidad para realizar las pruebas. Ya que es aquí donde se van a definir las actividades, responsables y el calendario para las mismas. Por esto se convierte en una necesidad la confección del mismo, sea cual sea el producto a probar. En el siguiente epígrafe se mostrará dicho plan, además de destacar sus principales elementos y estructura.

2.4 Plan de prueba general

Un plan de prueba define el marco para determinar si los entregables del proyecto se están realizando en las maneras en las cuales se esperaba que se realizaran. Este establece los objetivos y la estrategia para la evaluación del proyecto que será sometido a pruebas. También identifica a participantes en el proceso, sus papeles, y el ambiente en el cual las pruebas serán conducidas. [5]
En el plan de pruebas definido para el sistema CMM, se identifican los elementos que serán probados, los recursos necesarios para hacer las pruebas, así como la estrategia de pruebas que se llevará a

cabo para lograr un buen diseño de casos de pruebas que permitan encontrar la mayor cantidad de posibles fallos del software en cuestión.

Estas mismas ideas se suelen agrupar diciendo que un caso de prueba consta de 3 bloques de información: **(2)**

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

Para ver el plan de pruebas realizado para el proyecto CCM dirigirse a: *anexos/ Plan de Pruebas v2.0.doc.*

El objetivo que tenga un plan de pruebas es imprescindible para el proyecto, ya que el que va a definir el alcance del mismo teniendo una visión más amplia de este, a continuación se mostrará cual es el objetivo de este plan de pruebas, realizado fundamentalmente para guiar todo el proceso de pruebas que se desarrollará a lo largo de este trabajo.

2.5 Estrategia de pruebas

En cierta manera la estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. Describe el enfoque y los objetivos generales de las actividades de prueba, define: niveles, tipos, técnicas (manual o automática) de prueba a utilizar en el proyecto. Además de, ¿qué criterios de éxito y culminación de la prueba serán usados?, y alguna que otras consideraciones especiales afectadas por requerimientos de recursos o que tengan implicaciones en la planificación. [1]
En esta estrategia de prueba debe quedar bien definida cuales serán los recursos necesarios para que las mismas sean llevadas a cabo con toda la eficacia posible. Ahora en el siguiente epígrafe quedará plasmada de una manera clara toda la estrategia que se llevará durante el proceso de proceso de pruebas enmarcado en este trabajo.

2.5.1 Pasos a seguir en las pruebas o estrategia

Teniendo en cuenta las características del producto CCM, es un sistema para la captura y catalogación de medias, que realizará el despliegue en Venezuela, se definen los siguientes escenarios para la aplicación de las pruebas.

Inicialmente las pruebas que se le aplicarán al producto son las de Función. Estas pruebas se centran en la validación de funciones del destino de la prueba. Los niveles al cual se llevará a cabo son unidad e integración, asegurando que funcionan adecuadamente. Es aquí donde se le realizarán el diseño de casos de prueba de caja blanca a los casos de usos seleccionados, utilizando la técnica del camino básico. Esta técnica no se les aplicará a todos los CU con los que cuenta el proyecto CCM, pues esta práctica de pruebas requiere mucho tiempo para su ejecución. Si bien es cierto que entre más detalladas sean las pruebas que se le apliquen al producto, mayor será el grado de calidad que tendrá el mismo, hay que tener en cuenta el esfuerzo que requiere cada una de estas pruebas. Con esta técnica, por cada caso de prueba es necesaria la construcción de un grafo que represente todas las variantes que tiene el algoritmo implementado para darle solución al caso de uso. Ahora se demostrará mediante el método de Análisis de Puntos de Casos de Uso que permite calcular la estimación del esfuerzo en proyectos basados en casos de uso, que no será posible realizar el diseño y aplicación de todos los casos de prueba, tanto de caja blanca como de caja negra a todos los casos de uso del proyecto CCM en el tiempo especificado por este trabajo.

Actividad Porcentaje Horas	Hombre	Actividad Porcentaje Horas
Análisis	10.00%	755,496
Diseño	20.00%	1511
Programación	40.00%	3022
Pruebas	**15.00%**	**1133.244**
Sobrecarga (otras	15.00%	1133.244

Tabla 6: Criterio de evaluación en porcentaje. (15)

En la tabla número 11 queda claro que se necesitan 1133.244 horas para la etapa de pruebas. Teniendo en cuenta que para el diseño y aplicación de pruebas al proyecto CCM enmarcado para este trabajo solo se tienen 68 días laborables, se tendría que trabajar 16.2 h/días para poder realizar tanto el diseño como aplicación de todos los casos de uso con los que cuenta el proyecto (CCM), siendo esto una cifra humanamente casi imposible de lograr.

Luego de haber realizado todas las pruebas de Función, se deben ensamblar o integrar los módulos para formar el paquete de software completo. Aquí es donde se aplicarán las pruebas de Integridad, en este caso específicamente la integración ascendente, dado que los módulos se integran de abajo hacia arriba. Durante la integración, las técnicas que más prevalecen son las de diseño de caso de prueba de caja negra utilizando la técnica de partición equivalente, probando cada una de las entradas con las salidas esperadas y el nivel de prueba que se lleva a cabo es el de Integración.

Después de que el software se ha integrado se deben aplicar las pruebas de Función a nivel de sistema para verificar que cada elemento encaja de forma adecuada y que se alcanza la

funcionalidad y el rendimiento total del sistema utilizando pruebas de caja negra empleando la técnica de partición equivalente.

Con el fin de que se garantice una fiabilidad por parte del usuario se aplicarán las pruebas de Estructura, estas se centran en la evaluación de la adherencia del destino de la prueba a su diseño y formación. Con ella se garantiza que todos los enlaces están conectados, se muestra el contenido adecuado y no que no exista ningún contenido huérfano. Los niveles a los cuales se llevarán a cabo son integración y sistema.

Luego de mencionar toda la estrategia a seguir para la ejecución de las pruebas, es necesario tener en cuenta cuales son los recursos que hay que tener para que estas pruebas se lleven a cabo satisfactoriamente. A continuación se mostrarán dichos elementos.

1.6 Script de pruebas

Para la realización de las pruebas se hace necesario contar con algunos recursos tangibles dentro de los cuales están como mínimo:

Recursos de software:

Especificaciones de Software:

- Sistema Operativo: GNU/Linux Ubuntu.
- Servidor Web: Apache.
- Servidor de Base de Datos: PostgreSQL.
- Servidor Streaming: VLC.

Especificaciones de Hardware

Servidor	Procesador	Memoria RAM	Disco Duro	Tarjeta de Red	Tarjeta de Video
Polo	Intel (R) Core (TM) 2 Duo CPU E4500 @ 2.20 GHz (2 CPUs)	1 giga[8]byte (GB)	120 Gb	Intel (R) 82566DC gigabit Network Connection	Intel (R) G965 Express Chipset Family

Tabla 7: Especificaciones de hardware

Luego de mencionar estos recursos es necesario que se lleve a cabo un proceso de selección de los casos de uso [9](CU) a los cuales se les va a realizar las pruebas. Esta selección se llevará

[8] *Unidad de medida informática cuyo símbolo es el GB.*

[9] *Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas.*

a cabo utilizando un criterio de evaluación, con el objetivo fundamental de que esta elección de CU sea la que verdaderamente defina la calidad del producto. En el siguiente epígrafe se mostrará cómo es que se llevó a cabo todo este proceso.

1.7 Casos de Usos seleccionados para el proceso de pruebas

El Sistema de Captura y Catalogación de Medias (CCM), cuenta con 33 casos de uso repartidos por los 4 subsistemas antes mencionado. A continuación se muestra una tabla donde están recogidos estos casos de uso, separados por los módulos a los cuales pertenecen.

Tabla de casos de Usos de proyecto CCM

Subsistemas	Casos de Usos (CU)	Prioridad	CN	CB
Administración	Gestionar Rol	Crítico	X	
	Gestionar Usuarios	Crítico	X	
	Autenticar Usuario	Crítico	X	X
	Gestionar Planificación.	Crítico	X	
	Administrar Procesos	Crítico		
	Gestionar Servidor de Medias	Crítico	X	
	Gestionar configuración de Reportes	Crítico	X	X
	Visualizar Reporte	Crítico	X	
	Gestionar estación de trabajo	Crítico		
	Gestionar Local	Crítico		
Audio	Autenticar Usuario	Crítico		
	Gestionar Archivo	Crítico		
	Grabar Audio	Crítico	X	
	Grabación Automática	Crítico	X	
	Grabación Manual	Crítico	X	
	Reproducir Audio	Crítico	X	X
	Transcribir Audio	Crítico	X	X
Catalogación	Autenticar Usuario	Crítico	X	
	Buscar Medias	Crítico	X	
	Visualizar Reportes	Secundario		
	Exportar media	Secundario	X	
	Reproducir Medias	Crítico	X	X
	Editar Video	Crítico	X	X
	Catalogar Medias	Crítico	X	
Video	Gestionar Video	Crítico	X	
	Realizar captura de video	Crítico	X	
	Digitalizar	Crítico	X	
	Realizar Transcodificación	Crítico		X
	Registrar datos	Crítico	X	
	Transferir material	Crítico		
	Extraer fotogramas	Crítico		
	Generar Índices	Crítico		
	Autenticar Usuario	Secundario		X

Tabla 8: Casos de Usos de proyecto CCM

Para realizar esta selección de CU se tuvo en cuenta, la importancia que cada uno de ellos les ofrece al subsistema al cual pertenecen. Para ello se aplicó una entrevista a cada jefe de subsistema de proyecto CCM. La misma se hizo con el objetivo fundamental de que la selección de los CU fuera realizada por las personas que en realidad tienen el conocimiento óptimo acerca de todas las funcionalidades del proyecto (CCM). Con lo que se pudo determinar aquellos CU que tienen un carácter crítico dentro de un módulo, que en realidad definen la línea base de la arquitectura y que son los que en verdaderamente le van a dar cumplimiento a las funcionalidades primordiales que el cliente espera recibir. Al mismo tiempo se evaluaron las características que debían cumplir los CU de carácter crítico dentro de un módulo específico y se escogieron dentro de estos los que realmente son necesarios probar en esta etapa en la que se encuentra la construcción del producto.

Para ver los resultados de la entrevista ver *anexo: Entrevistas/Entrevista (nombre del módulo X).doc.*

Luego de haber tenido el siguiente criterio de elección se decide que: la selección de casos de uso a los cuales se les va a realizar el diseño de casos de prueba tanto de caja negra como de caja blanca son los que se muestran en la tabla 5: en las columnas CN (Caja negra), CB (Caja blanca), marcados con una X. Después de tener definidos estos CU es preciso analizar cuáles serán los requerimientos generales del producto. Por lo que se convierte en una necesidad mencionar las exigencias que serán objetos de prueba.

1.8 Diseño de casos de pruebas de Caja Negra

Con las pruebas de caja negra se comprueban las funcionalidades del sistema en general, para el desarrollo de las mismas, se aplica la técnica de partición equivalente que resulta efectiva a la hora de comprobar la validez de cada entrada en los diferentes módulos. A continuación se mostrarán los diseños de casos de pruebas realizados para el proyecto CCM.

1.8.1 Módulo de Audio

Nombre de CU: Grabar Audio

Descripción General: Mediante este caso de uso usuario puede capturar audio desde la aplicación, utilizando como fuente la línea de entrada. **(19)**

Condiciones de Ejecución: Que el usuario esté autenticado, que el usuario pertenezca a un rol con permisos de grabación de audio y que exista alguna actividad a grabar en la planificación de trabajo o se necesite realizar una grabación en un momento determinado. **(19)**

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Consultar planificaciones.	EC 1.1: Consultar planificaciones	El sistema muestra las planificaciones que cumplen con los parámetros introducidos por el usuario luego de seleccionar una planificación el sistema muestra la interfaz de grabación de audio, que muestra los datos de la planificación seleccionada (Nombre, Hora de inicio, Hora de Fin, Local y Actividad), muestra además información sobre el sistema (espacio disponible y estado). Además de dicha información el sistema ofrece la posibilidad de seleccionar una de dos opciones: “Grabación Automática” y “Grabación Manual”.
	EC 1.2: Filtros incorrectos.	El sistema no encuentra ninguna planificación que cumplen con los parámetros introducidos por el usuario, informa al usuario del problema ocurrido y finaliza el CU.
SC 2: Grabar Audio	EC 2.1: Grabar Audio exitosamente.	El sistema verifica la validez de los datos introducidos, y en caso de serlo habilita el resto de los controles de la interfaz y deshabilita los controles que recogen los datos introducidos por el usuario. Una vez habilitados los controles el sistema continúa según establece el Caso de Uso Grabación Manual.
	EC 2.2: Grabar Audio falla.	El sistema verifica la no validez de los datos introducidos, y no se habilita el resto de los controles de la interfaz. Una vez no habilitados los controles el sistema muestra un mensaje informando que los datos no son válidos finalizando el CU.

Tabla 9: Diseño de CN caso de uso Grabar Audio.

Nombre de CU: Grabación Automática

Descripción General: Mediante este caso de uso usuario puede capturar audio desde la aplicación, a partir de una planificación existente es decir de forma automática se iniciará la captura de audio, teniendo como datos aquellos establecidos en la planificación. **(19)**

Condiciones de Ejecución: Que el usuario esté autenticado, que el usuario pertenezca a un rol con permisos de grabación de audio y que exista alguna planificación de grabación de audio. **(19)**

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC1: Pausar/Continuar Grabación.	EC 2.1: Pausar/Continuar Grabación opción pause	El sistema establece en el Panel de Información del sistema el Estado: “En pausa”, y cambia el texto del botón Pausar por “Continuar”. Además el sistema deja de capturar audio al fichero creado.
	EC 2.1: Pausar/Continuar Grabación opción continuar.	El sistema establece en el Panel de Información del sistema el Estado: “Grabando”, y cambia el texto del botón Pausar por “Pausar”. Además el sistema continúa la capturar audio al fichero creado.
SC 2: Comenzar Grabación.	EC 3.1: Comenzar Grabación.	El sistema inicia la captura de audio desde la línea de entrada, estableciendo como Estado en el Panel de Información del Sistema el Estado: “Grabando”, que muestra entre paréntesis el tamaño del fichero creado a partir del audio capturado en la carpeta destino especificada y el tiempo transcurrido desde el inicio de la grabación en el formato HH:MM: SS.
SC 3: Terminar grabación.	EC 4.1: Terminar grabación exitosamente.	El sistema crea un fichero WAV a partir del fichero capturado y muestra mediante la barra de progreso “Creación de Archivo WAV” el porciento de creación del fichero. Una vez culminado este proceso el sistema crea un fichero MP3 a partir del fichero WAV obtenido en el paso anterior y muestra mediante la barra de progreso “Transcodificación a MP3” el porciento de creación del fichero. Una vez culminado el paso anterior el sistema inicia la transferencia del fichero MP3 al servidor seleccionado y muestra mediante la barra de progreso “Transferencia al Servidor” el porciento de creación del fichero.

	EC 4.1: Terminar grabación falla.	El sistema detecta un error al crear el fichero WAV a partir del fichero capturado y muestra un mensaje de error informando del error ocurrido o detecta un error al crear el fichero MP3 a partir del fichero WAV obtenido en el paso anterior. Puede ocurrir también que se produzca un error en la transferencia del fichero MP3 al servidor seleccionado.

Tabla 10: Diseño de CN caso de uso Grabación Automática

1.8.2 Módulo Administración

Nombre de CU: Gestionar Rol (20)

Descripción General: El caso de uso se inicia cuando el administrador solicita insertar, modificar o eliminar un rol en la Base de Datos o un permiso asociado a uno de los roles.

Condiciones de Ejecución: Que el administrador se haya autenticado.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Insertar Rol	EC 1.1: Solicita insertar Rol exitosamente.	El sistema comprueba la validez de la información del nuevo rol que se desea insertar. Se busca el rol en la base de datos (BD) para verificar que no exista. De no ocurrir ningún problema en todo el flujo de sucesos se almacena en la BD los datos del nuevo rol y concluye el CU.
	EC 1.2: insertar Rol falla	El sistema comprueba la no validez de la información o se verifica que el rol a insertar ya existe en la BD mostrándole un mensaje al usuario explicando que no es posible la inserción de nuevo rol por el motivo encontrado.
SC 2: Modificar Rol	EC 2.1: Modificar Rol exitosamente.	El sistema verifica la existencia del rol especificado en la BD. Muestra información del rol que se desea modificar. Luego comprueba la validez de los nuevos datos del rol. Se actualiza en la BD los datos del rol modificado terminando el CU.
	EC 2.2: Modificar Rol falla.	El sistema comprueba la no validez de la información y muestra un mensaje al usuario explicando que no es posible la modificación del Rol.

SC 3: Eliminar Rol	EC 3.1: Eliminar Rol exitosamente.	El sistema muestra la información del rol que se desea eliminar pide confirmación de la acción de eliminación del Rol establecido, y ante la respuesta positiva borra de la base de datos la información del Rol seleccionado finalizando el CU.

Tabla 11: Diseño de CN caso de uso Gestionar Rol

Nombre de CU: Gestionar Usuarios (20)

Descripción General: El caso de uso permite insertar, modificar o eliminar usuarios, registrándose los cambios en la Base de Datos.

Condiciones de Ejecución: Que el administrador se haya autenticado. Que exista al menos un rol en la Base de Datos.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Insertar Usuario	EC 1.1: Insertar Usuario exitosamente	El sistema comprueba la validez de la información, verifica que usuario no exista en la BD, de comprobar que no exista lo añade a la BD concluyendo el CU.
	EC 1.2: Insertar Usuario falla	El sistema comprueba que existen errores en la información del usuario o que el mismo ya se encuentra en la BD y muestra un mensaje al usuario explicando que no es posible la inserción de nuevo usuario.
SC 2: Modificar Usuario	EC 2.1: Modificar Usuario exitosamente	El sistema verifica existencia en la BD del usuario especificado. Muestra información del usuario. Verifica que esté correcta la información del usuario a modificar. Luego verifica que el usuario no se encuentre autenticado, de no estarlo se actualiza el usuario en la BD y se finaliza el caso de uso.
	EC 2.2: Modificar Usuario falla.	El sistema verifica la no existencia en la BD del usuario especificado. Comprueba que existen errores en la información del usuario a modificar o que el usuario esta autenticado mostrando un mensaje al usuario informando del problema ocurrido por lo que no se modifica el usuario.
SC 3: Eliminar Usuario	SC 3: Eliminar Usuario exitosamente	El sistema pide confirmación de la acción de eliminación del usuario establecido, y ante la respuesta positiva verifica que el usuario no se encuentre autenticado, de no estarlo borra de la BD la información del usuario seleccionado finalizando el CU.

Tabla 12: Diseño de CN caso de uso Gestionar Usuario

1.8.3 Módulo Video

Nombre de CU: Gestionar Video **(21)**

Descripción General: Mediante este caso de uso el usuario de captura puede subir ficheros al servidor y hacer copias de estos fuera del servidor.

Condiciones de Ejecución: Para la ejecución de este caso de uso se necesita que exista un dispositivo disponible para realizar la captura o un fichero para almacenar en el servidor. Para comenzar a ejecutar este caso de uso debemos estar en la interfaz correspondiente al proceso de gestionar video.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Subir material analógico.	EC 1.1: Subir material analógico.	El usuario de captura solicita la opción de subir un material analógico al servidor de medias. Se cargan los dispositivos de captura del ordenador y se procede a la digitalización (Ver Caso de Uso Digitalizar). Finalizando el caso de uso.
SC 2: Subir material digital.	EC 2.1: Subir material digital con el formato correcto.	El usuario de captura solicita la opción de subir un material digital al servidor. Para subir un material al servidor es necesario que este esté en el formato correcto. El sistema comprueba que el formato del fichero MPEG2. Guarda en el servidor de media y registra los datos en el servidor de BD. (Ver Casos de Usos Transferir material y Registrar datos).
	EC 2.2: Subir material digital con el formato incorrecto.	El usuario de captura solicita la opción de subir un material digital al servidor. Como este material no está en el formato MPG2, se transcodifica el mismo antes de subirlo al servidor. Finalizando el CU. (Ver caso de uso Realizar Transcodificación).
SC 3: Hacer Copia.	EC 3.1 Realizar Copia.	El usuario introduce en el campo de búsqueda los parámetros necesarios para realizar la búsqueda. Estos se muestran y este debe seleccionar cual desea copiar posibilitándose la opción de selección de un destino. Posteriormente procede a realizar la copia.

	EC 3.2 No se encuentra ningún material en el servidor con los criterios de búsqueda especificados.	Después de establecer los criterios de búsqueda, si no hay ningún material en el servidor que cumpla con estos, el sistema debe mostrar un mensaje notificando al usuario lo sucedido.

Tabla 13: Diseño de CN caso de uso Gestionar Video

Nombre de CU: Realizar captura de video. **(21)**

Descripción General: El caso de uso se inicia siguiendo los tiempos definidos por la planificación descrita por el servidor de BD. Realiza una conversión del material a partir de las especificaciones del formato de video al que se desea transcodificar.

Condiciones de Ejecución: Debe existir una planificación para ser capturada en la PC de captura especificada.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Realizar captura de video	EC 1.1: Capturar un material	El sistema lee desde el servidor de BD los datos pertinentes a la captura de video en dicha estación (canal, dispositivo, fecha, tiempo inicio, tiempo fin, nombre del material) para ejecutar la captura de video a partir de la fecha planteada, siguiendo los tiempos leídos y transcodificar el material (ver descripción del Caso de Uso Realizar transcodificación).

Tabla 14: Diseño de CN caso de uso Realizar captura de video

1.8.4 Módulo Catalogación

Nombre de CU: Autenticar Usuario **(22)**

Descripción General: El caso de uso se inicia cuando el actor introduce su usuario y contraseña para acceder al sistema, los mismos se verifican contra la base de datos. Finaliza cuando se habilita la entrada al usuario con los permisos asignados a este o se deniega su acceso.

Condiciones de Ejecución:

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Autenticar Usuario.	EC 1.1: Autenticar un usuario válido con la contraseña correcta. (Un usuario es válido cuando se encuentra	Se introducen los datos requeridos para la autenticación de un usuario que se encuentre registrado en la BD: usuario y contraseña. Se le permite el acceso a las funcionalidades del sistema según sus permisos.
	EC 1.2: Autenticar un usuario no válido.	Se introducen los datos requeridos para la autenticación de un usuario que no se encuentre registrado en la base de datos: usuario y contraseña. No se le permite el acceso al sistema, se muestra un mensaje de error: "Usuario o contraseña incorrecto" y se le da la oportunidad de volver a autentificarse.
	EC 1.3: Autenticar un usuario válido con la contraseña incorrecta.	Se introducen los datos requeridos para la autenticación de un usuario que se encuentre registrado en la base de datos: usuario y contraseña, en este caso con una contraseña incorrecta. No se le permite el acceso al sistema, se muestra un mensaje de error: "Usuario o contraseña incorrecto" y se le da la oportunidad de volver a autentificarse.

Tabla 15: Diseño de CN caso de uso Autenticar Usuario

Nombre de CU: Buscar Medias **(22)**

Descripción General: El caso de uso se inicia cuando el usuario introduce los parámetros necesarios para localizar la media que necesita en el servidor. Finaliza con el listado de los materiales que cumplan con los criterios solicitados, puede ser ninguno, uno o varios.

Condiciones de Ejecución: El usuario debe estar autenticado y tener permisos para utilizar la funcionalidad.

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad

SC 1 Realizar Búsqueda Básica.	EC 1.1: Criterios de búsqueda introducidos correctamente.	El sistema busca en la BD las referencias que cumplan con los criterios señalados por el usuario. El sistema muestra en una tabla el resultado de la búsqueda finalizando el caso de uso.
	EC 1.2: Criterios de búsqueda introducidos incorrectamente.	El sistema detecta que ocurrió un error en los criterios introducidos, se informa al usuario mediante un mensaje de error finalizando el CU.
	EC 1.3: Fallo en la conexión a la base de datos.	El sistema detecta que hubo un problema de conexión con la BD se informa al usuario mediante un mensaje de error finalizando el CU.

Tabla 16: Diseño de CN caso de uso Buscar Medias

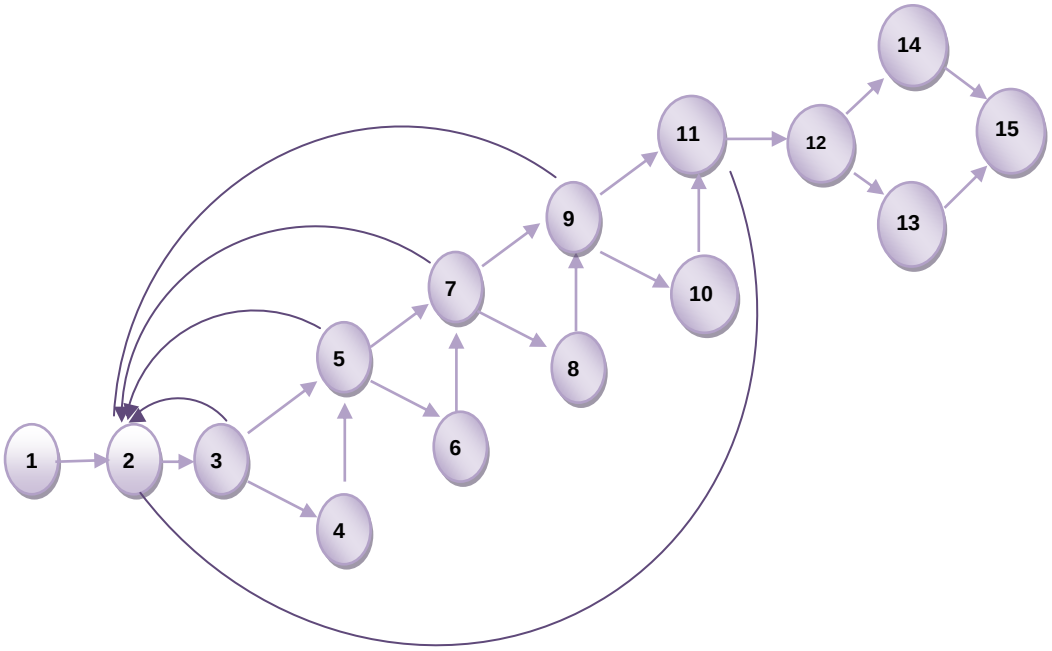
1.9 Diseño de casos de prueba Caja Blanca

Con las pruebas de caja blanca el ingeniero de software puede comprobar que se verifiquen las funciones internas de un módulo. La técnica que se utilizará para la construcción de los casos de prueba es la del camino básico. A continuación se mostrarán los diferentes diseños realizados para cada módulo del proyecto CCM.

2.9.1 Módulo Audio

CU: Reproducir Audio

Paso 1: Dibujar el grafo de Flujo asociado a partir del código fuente:



Paso 2: Cálculo de complejidad ciclomática de Reproducir Audio

V (G)= Número de Aristas – Número de Nodos + 2 = 11.

Paso 3: Caminos básicos:

CB1: 1-2-3-4-5-6-7-8-9-10-11-12-13-15

CB2: 1-2-5-7-9-11-12-13-15

CB3: 1-2-5-7-9-11-12-14-15

CB4: 1-2-11-13-15

CB5: 1-2-11-14-15

CB6: 1-2-3-2-11-13-15

Paso 4: Caso de prueba para el camino básico.

CB1: 1-2-3-4-5-6-7-8-9-10-11-12-13-15

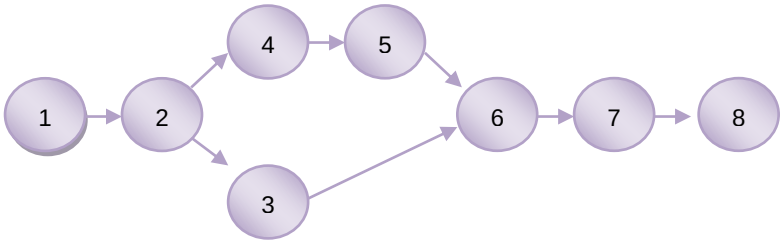
Entrada: Reproducir = Activado, Detener = Desactivado, Adelantar = Desactivado, Atrasar = Desactivado, Volumen = 50%.

Resultado Esperado: El sistema reproduce el archivo.

Condiciones: Que el usuario esté autenticado, pertenezca a un rol con permisos de reproducción de Audio e inicie el caso de uso Gestionar Archivo en su sección consultar archivo o el CU Transcribir Audio.

CU: Transcribir Audio.

Paso 1: Dibujar el grafo de Flujo asociado a partir del código fuente:



Paso 2: Cálculo de complejidad ciclomática de Trancodificar Audio

V (G)= Número de Aristas – Número de Nodos + 2 = 2

Paso 3: Caminos básicos:

CB1: 1-2-3-6-7-8

CB2: 1-2-4-5-6-7-8

Paso 4: Caso de prueba para el camino básico.

CB1: 1-2-3-6-7-8

Entrada: Reproducir = Activado, Detener = Desactivado, Adelantar = Desactivado, Atrasar = Desactivado, Guardar sin terminar = Desactivado, Terminar Transcripción = Desactivado.

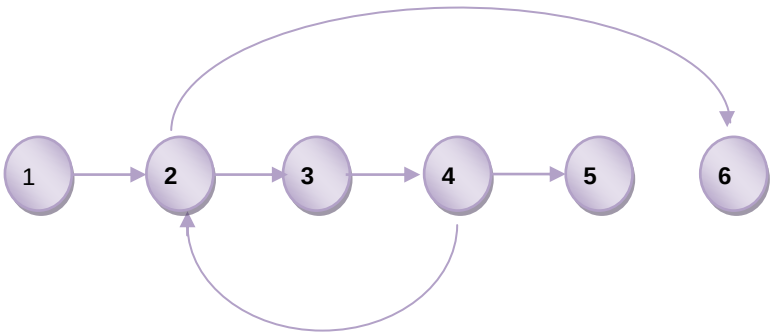
Resultado Esperado: El sistema muestra una interfaz con un editor de texto para transcribir y un reproductor de audio para escuchar el audio que se transcribe.

Condiciones: Que el usuario esté autenticado, pertenezca a un rol con permisos de transcripción y que exista audio no transcrito en el servidor.

2.9.2 Módulo Administración

CU: Autenticar Usuario

Paso 1: Dibujar el grafo de Flujo asociado a partir del código fuente:



Paso 2: Cálculo de complejidad ciclomática de Autenticar Usuario

V (G)= Número de Aristas – Número de Nodos + 2 = 2

Paso 3: Caminos básicos:

CB1: 1-2-3-4-2-6

CB2: 1-2-6

Paso 4: Caso de prueba para el camino básico.

CB1: 1-2-6

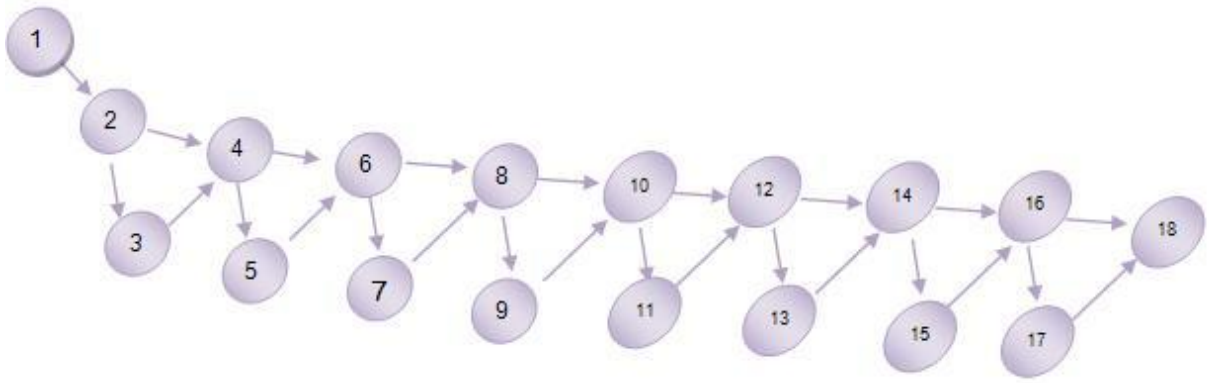
Entrada: Usuario = Gilberto, Contraseña = *********

Resultado Esperado: Tras la inserción de los datos correspondientes como usuario y contraseña, el proceso de autenticación se lleva con éxito.

Condiciones: Que el usuario exista en la base de datos

CU: Generar Reportes

Paso 1: Dibujar el grafo de Flujo asociado a partir del código fuente:



Paso 2: Cálculo de complejidad ciclomática de Generar Reportes

V (G)= Número de Aristas – Número de Nodos + 2 = 9

Paso 3: Caminos básicos:

CB1: 1-2-4-6-8-10-12-14-16-18

CB2: 1-2-3-4-6-8-10-12-14-16-18

CB3: 1-2-4-5-6-8-10-12-14-16-18

CB4: 1-2-4-6-7-8-10-12-14-16-18

CB5: 1-2-4-6-8-9-10-12-14-16-18

CB6: 1-2-4-6-8-10-11-12-14-16-18

CB7: 1-2-4-6-8-10-12-13-14-16-18

CB8: 1-2-4-6-8-10-12-14-15-16-18

CB9: 1-2-4-6-8-10-12-14-16-17-18

Paso 4: Caso de prueba para el camino básico.

CB1: 1-2-4-6-8-10-12-14-16-18

Entrada: Nombre configuración = Estudiar, Configuración semanal = Martes, Tipo de reporte= Operaciones realizadas, Hora = 20.30.00

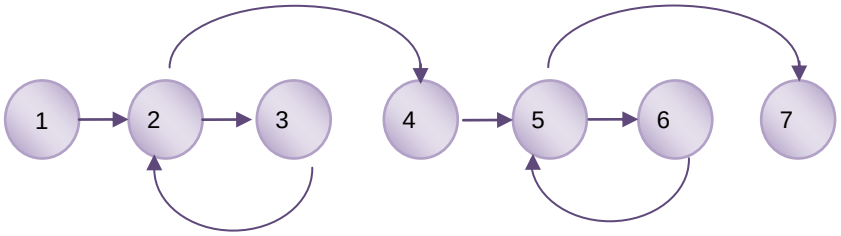
Resultado Esperado: El sistema generará reportes cada un instante de tiempo definido y según especificaciones realizadas en la configuración del reporte, se guarda el reporte en la Base de datos y se finaliza el caso de uso.

Condiciones: Que un usuario este autenticado como Administrador.

2.9.3 Módulo Catalogación

CU: Editar Video

Paso 1: Dibujar el grafo de Flujo asociado a partir del código fuente:



Paso 2: Cálculo de complejidad ciclomática de Editar Video.

V (G)= Número de Aristas – Número de Nodos + 2 = 3

Paso 3: Caminos básicos:

CB1: 1-2-4-5-7

CB2: 1-2-3-2-4-5-6-5-7

CB3: 1-2-4-5-6-5-7

Paso 4: Caso de prueba para el camino básico.

CB1: 1-2-4-5-7

Entrada: Tiempo inicio = 01.20.30 Tiempo Fin = 12.00.00

Resultado Esperado: El sistema guarda la referencia del segmento deseado en la base de datos.

Condiciones: Es necesario haber hecho una búsqueda previa en la base de datos para obtener el listado de las medias disponibles. Que esté autenticado y tenga permisos para utilizar la funcionalidad.

CU: Reproducir Medias.

Paso 1: Dibujar el grafo de Flujo asociado a partir del código fuente:

Paso 2: Cálculo de complejidad ciclomática de Reproducir Medias.

V (G)= Número de Aristas – Número de Nodos + 2 = 1

Paso 3: Caminos básicos.

CB1: 1

Paso 4: Caso de prueba para el camino básico.

CB1: 1

Entrada: Conexión Streaming = True, Conexión a la BD = True.

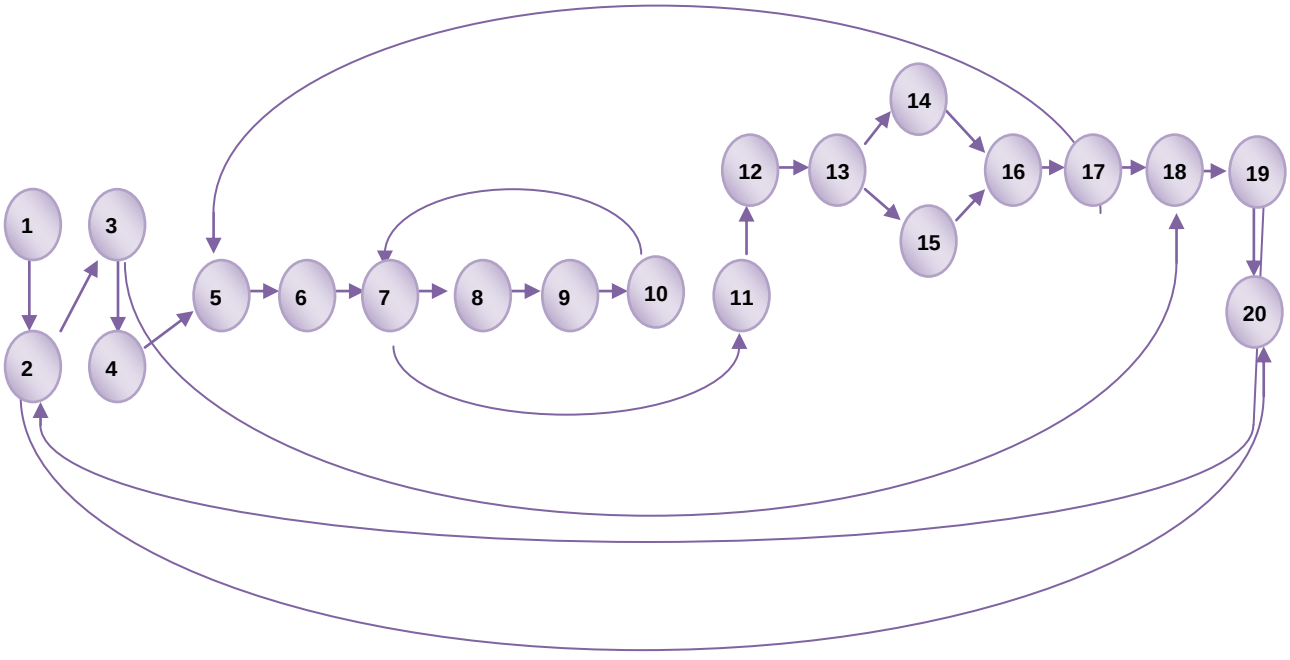
Resultado Esperado: El sistema reproduce la media satisfactoriamente.

Condiciones: Es necesario haber hecho una búsqueda previa en la base de datos para obtener el listado de las medias disponibles. Además que esté autenticado y tenga permisos para utilizar la funcionalidad.

2.9.4 Módulo Video

CU: Autenticar Usuario

Paso 1: Dibujar el grafo de Flujo asociado a partir del código fuente:



Paso 2: Cálculo de complejidad ciclomática de Autenticar Usuario.

V (G)= Número de Aristas – Número de Nodos + 2 = 7

Paso 3: Caminos básicos.

CB1: 1-2-3-4-5-6-7-8-9-10-7-11-12-13-14-16-17-18-19-20

CB2: 1-2-3-4-5-6-7-8-9-10-7-11-12-13-15-16-17-18-19-20

CB3: 1-2-20

CB4: 1-2-3-18-19-20

CB5: 1-2-3-4-5-6-7-11-12-13-15-16-17-18-19-20

CB6: 1-2-3-4-5-6-7-8-9-10-7-11-12-13-15-16-17-5-6-7-11-12-13-15-16-17-18-19-20

CB7: 1-2-3-4-5-6-7-8-9-10-7-11-12-13-15-16-17-18-19-2-3-4-5-6-7-11-12-13-14-16-17-18-19-20

Paso 4: Caso de prueba para el camino básico.

CB1: 1-2-3-4-5-6-7-8-9-10-7-11-12-13-14-16-17-18-19-20

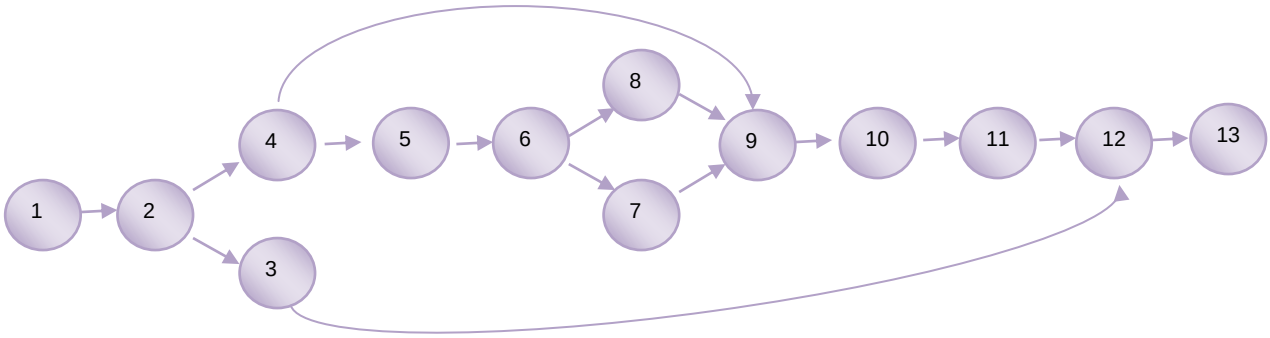
Entrada: Usuario = gvelazquez**,** contraseña = *******

Resultado Esperado: Se le permite el acceso al sistema (mostrando la interfaz correspondiente) con los permisos que tiene asignado el rol al que pertenezca.

Condiciones: No tiene.

CU: Realizar transcodificación.

Paso 1: Dibujar el grafo de Flujo asociado a partir del código fuente:



Paso 2: Cálculo de complejidad ciclomática de Realizar transcodificación.

V (G)= Número de Aristas – Número de Nodos + 2 = 4

Paso 3: Caminos básicos.

CB1: 1-2-3- 12-13

CB2: 1-2-4-9-10-11-12-13

CB3: 1-2-4-5-6-8-9-10-11-12-13

CB4: 1-2-4-5-6-7-9-10-11-12-13

Paso 4: Caso de prueba para el camino básico.

CB1: 1-2-3- 12-13

Entrada: Material = Corazón valiente

Resultado Esperado: Se obtiene un material codificado en un directorio local.

Condiciones: Se debe haber comenzado una captura de video, un proceso de digitalización o existir un material en un formato no deseado.

1.10 Pruebas de Estructura

Este tipo de prueba no cuenta con un diseño definido para su ejecución, pero su práctica es muy eficiente en los productos informáticos, ya que son pruebas que se centran en la evaluación de la adherencia del destino de la prueba a su diseño y formación. Normalmente, esta prueba se realiza en aplicaciones habilitadas para web pero también pueden aplicarse a aplicaciones de escritorio y garantiza que todos los enlaces están conectados, se muestra el contenido adecuado y no hay ningún contenido huérfano.

La prueba de estructura se implementa y ejecuta para verificar que todos los enlaces (estáticos o activos) están conectados correctamente. Estas pruebas incluyen:

- ✓ **Verificación de que se muestra el contenido correcto (texto, gráficos, etc.) de cada enlace.** Se utilizan diferentes tipos de enlaces para hacer referencia al contenido de destino de las aplicaciones. Deben comprobarse todos los enlaces para garantizar que se muestra el contenido de destino correcto a los usuarios. **(8)**
- ✓ **Comprobación de que no hay enlaces rotos.** Los enlaces rotos son los enlaces cuyo contenido de destino no se puede encontrar. Los enlaces pueden estar rotos por muchos motivos, incluidos el desplazamiento, la eliminación o el cambio de nombre de los archivos de contenido de destino. Los enlaces también pueden estar rotos debido a un uso incorrecto de la sintaxis, incluidos los dos puntos, las barras inclinadas o las letras que falten. **(8)**
- ✓ **Verificación de que no hay contenido huérfano.** El contenido huérfano son los archivos que no tienen enlaces "de entrada" en la aplicación es decir, no se puede acceder a su contenido ni se puede presentar. Debe investigarse con cuidado el contenido huérfano para determinar la causa: **(8)**
    1. ¿Es huérfano porque realmente ya no es necesario?
    2. ¿Es huérfano debido a un enlace roto?

Una vez que se haya determinado la causa, debe llevarse a cabo la acción adecuada, como eliminar el archivo de contenido, reparar el enlace roto o ignorar el huérfano, respectivamente.

1.11 Conclusiones

En este capítulo se describió de forma clara las características del proyecto CCM, siendo este el producto que fue objeto de pruebas. A partir de esto, se describieron los diferentes módulos del mismo, con el objetivo fundamental de conocer las principales funcionalidades que fueron sometidas a prueba. Fue aquí donde se definió la estrategia de prueba, los recursos necesarios y plan de pruebas que se llevarían a cabo, garantizando una mayor claridad, destreza y rapidez durante toda la ejecución de las pruebas. También se llevó a cabo un proceso de selección de los casos de uso con los cuales se desarrollaron las pruebas, favoreciendo así la realización de los diseños de pruebas correspondientes, lo que permitió dar cumplimiento a los objetivos de este trabajo. También se confeccionaron todos los diseños de casos de pruebas tanto de caja negra como de caja blanca, con el objetivo esencial de que el proceso de prueba fuera de la manera más completa posible y facilitando la detección de la mayor cantidad de defectos posibles.

CAPÍTULO 3

Resultados

2.6 Introducción

Luego de haber llevado a cabo todo el proceso de pruebas correspondiente al producto CCM. En este capítulo se hace indispensable documentar todos los resultados obtenidos, con el objetivo de realizar un análisis de los errores encontrados durante todo el proceso de prueba. Posibilitando así tener un registro de estos y corregirlos en el menor tiempo posible, garantizando que el producto CCM sea entregado al usuario final con la menor cantidad de defectos posibles. Se hace referencia a los datos de pruebas que fueron utilizados para realizar las diferentes combinaciones de pruebas posibles al sistema CCM. También se documenta recomendaciones que se podrían tener en cuenta para posteriores versiones del producto.

2.7 Datos de Prueba

Los datos de prueba que se utilizaron para las pruebas al sistema CMM cubren los posibles valores de cada parámetro basado en los requerimientos de la base de datos. Para llevar a cabo la ejecución de casos de prueba de caja negra se toma los siguientes criterios.

1. El caso de prueba se ejecuta una vez por cada combinación de valores.
2. Se escogen 3 valores de cada clase de equivalencia (conjunto de valores que deberán ser tratados iguales).

A continuación se muestra una tabla donde se recogen todas las combinaciones de datos utilizadas para las pruebas, esto garantiza que las pruebas sean lo más completas posibles.

Juegos de datos utilizados en el sistema CCM para las pruebas de Caja Negra

Campo de la interfaz	Datos Válidos	Datos Inválidos
Nombre	Cualquier combinación de Letras (Gilberto, Anay, Sonia, Isaac) hasta 30 caracteres.	Números(1.263512099.411)
Descripción	Cualquier combinación de Letras (Gilberto, Anay, Sonia, Isaac), hasta 30 caracteres.	Números(1.263512099.411)
Permisos	Activado	Desactivado

Nueva descripción	Cualquier combinación de Letras (Gilberto, Anay, Sonia, Isaac,), hasta 30 caracteres.	Números (1.263512099.411).
Permisos	Selección de un elemento	No selecciono ningún elemento.
Usuario	Cualquier combinación de Letras (Gilberto, Sonia, Isaac,), hasta 30 caracteres.	Números (1.263512099.411).
Contraseña	Contraseña > =8	Contraseña < =8.
Campo de textos	Cualquier combinación de letras.	Números (0120.4421) caracteres especiales (¡@#$%^&**), combinación de ambos (1289@%^#&).
Los CheckBox	Se marcan.	No se marca ninguno.
Listas desplegables	Se selecciona un elemento.	No se selecciona ningún elemento.
Campo de texto	Se introducen cualquier combinación de letras hasta 30 caracteres ejemplo (Gilberto, Anay, Sonia, Isaac, Velázquez).	Números (0120.4421) caracteres especiales (¡@#$%^&**), combinación de ambos (1289@%^#&), se dejan campos vacios.
Destino	Se selecciona un lugar válido y que cuente con el espacio disponible.	Se selecciona una ruta inválida o una ruta que no cuente con la capacidad que necesita.
Volumen	Se establece el volumen en 80%.	Se establece el volumen en 0%.
Comenzar	Se activa el botón.	No se activa el botón.
Menú	Se seleccionan todas las opciones, editar fuente, tamaño y color.	No se selecciona ninguna opción.

Tabla 17 : Juegos de datos utilizados para las pruebas

Después de concluir el proceso de pruebas, es necesario documentar cada resultado obtenido con el objetivo principal de que estos queden plasmados por escrito. A continuación se muestran los resultados obtenidos después de llevar a cabo todas las pruebas correspondientes al producto CCM.

2.8 Resultados obtenidos de aplicar pruebas de caja negra

Luego de haber llevado a cabo todo el proceso de pruebas es necesario documentar los defectos encontrados, con el fin de que se tenga un control de los mismos para que puedan ser corregidos

por el equipo de desarrollo en el tiempo especificado. A continuación se mostrarán los principales errores encontrados en cada módulo del proyecto CCM.

2.8.1 Módulo Administración

Funcionalidad: Gestionar Rol

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Insertar Rol exitosamente		Se muestra un mensaje al administrador informándole que la acción ha sido llevada a cabo con éxito.	Se inserta el usuario satisfactoriamente.	La interfaz que se presenta es poco amigable.
	Insertar Rol falla	Muestra mensaje advirtiendo el error en la información.	Se muestra el mensaje satisfactoriamente.	La interfaz que se presenta es poco amigable.
Modificar Rol exitosamente		Se actualiza el Rol en la Base de Datos (BD) y se muestra mensaje que informando que la actualización se ha realizado con éxito.	Se modifican los datos satisfactoriamente.	La interfaz que se presenta es poco amigable.
	Modificar Rol falla	Se emite mensaje de error en la información que se desea modificar.	Se muestra el mensaje satisfactoriamente.	La interfaz que se presenta es poco amigable.
Eliminar Rol		Se actualiza el rol en la BD y se muestra mensaje que informe que la actualización se ha realizado con éxito.	Se elimina el rol satisfactoriamente.	La interfaz que se presenta es poco amigable.

Tabla 18: Resultados de las pruebas al CU Gestionar Rol

Funcionalidad: Gestionar Usuarios

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Insertar Usuario Exitosamente		Tras la aserción de los datos exigidos se muestra un mensaje al administrador informándole que la acción ha sido llevada a cabo con éxito.	Se inserta correctamente el usuario en la BD, y muestra el mensaje.	
	Insertar	Muestra mensaje advirtiendo al	Muestra un	

	Usuario falla	usuario el error que ocurrió con la información pertinente.	mensaje de error.	
Modificar Usuario Exitosamente		Se actualiza el usuario en la BD y se muestra mensaje que informe que la actualización se ha realizado con éxito.	Se actualiza correctamente el usuario en la BD y se informa con un mensaje.	
	Modificar Usuario falla	Se emite mensaje de error en la información que se desea modificar.	No se actualiza correctamente y se informa con un mensaje.	
Eliminar usuario exitosamente		Se elimina el usuario en la BD y se muestra el mensaje informando que la acción se ha realizado con éxito.	Se elimina correctamente el usuario en la BD y se informa con un mensaje.	

Tabla 19: Resultados de las pruebas al CU Gestionar Usuarios

2.8.2 Módulo Audio

Funcionalidad: Grabar Audio

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Consultar planificaciones		Se muestran las planificaciones que cumplen con los filtros seleccionados. Se puede seleccionar una de ellas y luego acceder a los controles de grabación.	Satisfactorio	
	Filtros incorrectos	El sistema emite un mensaje al usuario informando que existe un error en los filtros de búsqueda o que no se puede encontrar ninguna planificación.	Satisfactorio	

Tabla 20: Resultados de las pruebas al CU Grabar Audio

Funcionalidad: Grabación Automática

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones

Pausar/Continuar Grabación opción pause		El sistema deja de capturar audio al fichero creado.	No Satisfactorio	
Pausar/Continuar Grabación opción continuar		El sistema continúa la capturar audio al fichero creado.	No Satisfactorio	
Comenzar Grabación		El sistema inicia la captura de audio desde la línea de entrada, estableciendo como estado en el panel de Información del sistema el estado: “Grabando”.	Satisfactorio	
Terminar grabación exitosamente		El sistema termina el proceso de grabación y crea un fichero WAV a partir del fichero capturado creando a su vez un fichero MP3 a partir del fichero WAV obtenido en el paso anterior. Se inicia la transferencia del fichero MP3 al servidor seleccionado informando al usuario que la acción ha sido llevada a cabo satisfactoriamente.	No Satisfactorio	
	Terminar grabación falla	El sistema no termina el proceso de grabación y espera a que este sea finalizado por la acción del botón terminar.	No Satisfactorio	Debe de mostrar al usuario del error ocurrido.

Tabla 21: Resultados de las pruebas al CU Grabación Automática

2.8.3 Módulo Catalogación

Funcionalidad: Autenticar Usuario

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Autenticar un usuario válido		Se le permite el acceso al sistema (mostrando la interfaz	Satisfactorio.	La ventana que permite el acceso

con la contraseña correcta.		correspondiente) con los permisos que tiene asignado el rol al que pertenezca.		al sistema debe estar situada en el medio de la pantalla.
	Autenticar un usuario no válido.	El sistema no permite el acceso, Se muestra un mensaje de error: "Usuario incorrecto". Se ofrece la oportunidad de volver a introducir sus datos.	Satisfactorio.	
	Autenticar un usuario válido con la contraseña incorrecta.	El sistema no permite el acceso, se muestra un mensaje de error: "Contraseña incorrecta". Se ofrece la oportunidad de volver a introducir sus datos.	Satisfactorio.	

Tabla 22: Resultados de las pruebas al CU Autenticar Usuario

Funcionalidad: Buscar Media

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Criterios de búsqueda introducidos correctamente.		Se muestran todas las medias que cumplen con los criterios de búsqueda introducidos por el usuario.	Satisfactorio.	El campo de fecha debería ser del formato especificado ejemplo: Calendario.
	Criterios de búsqueda introducidos incorrectamente	Se informa al usuario mediante un mensaje de error finalizando el caso de uso.	Satisfactorio.	En el menú si se escoge la opción búsqueda avanzada mostrar la interfaz correspondiente.
	Fallo en la conexión a la base de datos.	Se informa al usuario mediante un mensaje de error. Finalizando el caso de uso.	No Satisfactorio.	No muestra el mensaje de error correspondiente.

Tabla 23: Resultados de las pruebas al CU Buscar Media

2.8.4 Módulo Video

Funcionalidad: Gestionar video

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Subir material digital con el formato correcto.		El sistema comprueba que el formato del fichero MPEG2, lo guarda en el servidor de medias y registra los datos en el servidor de BD. (Ver Casos de Usos Transferir material y Registrar datos.	No Satisfactorio	Esta funcionalidad no está implementada.
	Subir material digital con el formato incorrecto.	El sistema comprueba que el formato del fichero MPEG2, como esto no se cumple en este escenario, se transcodifica el material antes de ser almacenado en el servidor de medias. Luego se registran los datos en el servidor de BD. (Ver Casos de Uso Transferir material y Registrar datos).	Satisfactorio	
Realizar copia		Se muestran los materiales que existen en el servidor de medias que cumplen con los parámetros especificados por el usuario en la variable 1 (Criterio de búsqueda). El sistema realiza una copia del fichero en el servidor en la dirección indicada en la variable 2 (Especificar dirección de destino de la copia.) por el usuario, del material seleccionado en la variable 3 (Seleccionar material).	Satisfactorio	Se recomienda que no se muestre el mismo mensaje. El sistema mostrará un mensaje: Debe especificar el material que desea copiar.

No se encuentra ningún material en el servidor con los criterios de búsqueda especificados.		Si no se encuentra ningún material en el servidor de medias que cumpla con lo establecido en la variable1, la variable 3, estará vacía y se mostrará el mensaje: No se encuentran materiales con estos parámetros.	Satisfactorio	

Tabla 24: Resultados de las pruebas al CU gestionar video

2.9 Resultados de las pruebas de caja blanca

Las pruebas de Caja Blanca se ejecutaron utilizando una PC con sistema operativo Linux Ubuntu 9.4, con 1GB de RAM y un procesador Intel (R) Core (TM) 2 Duo CPU E4500 @2.20 GHz. Se corrió la aplicación donde estaban todas las interfaces generadas para cada uno de los módulos que se probaron. El sistema exigía una conexión a una base de datos que se encontraba en un servidor con las mismas características de hardware que las PC terminales y también con un servidor de streaming pues es necesario la reproducción de medias en tiempo real utilizando como servidor el VLC. A continuación se muestran los principales problemas encontrados.

Pruebas de caja blanca	Principales dificultades y errores encontrados.	Clasificación	Observaciones
Resultados de las pruebas de caja blanca.	En el caso de uso Transcribir Archivo Se transcribe el archivo pero si se le aplica un formato a la letra no hay forma de quitárselo.	No significativa	
	Acepta contraseñas menores de 7 caracteres en el caso de uso Autenticar Usuario del módulo de administración.	Significativa	La ventana del autenticar debe ser más pequeña.
	En el caso de uso Generar Reporte no tiene en cuenta el tipo de reporte.	Significativa	
	En el caso de uso Generar Reporte no tiene en cuenta a la hora de generar el mismo la	Significativa	Se deben de Señalar los campos significativos de la

	hora planificada.		interfaz
	Hay mensajes que no emiten ningún texto aclaratorio	Recomendación	
	No se tienen en cuenta los tratamientos de excepciones en los casos de uso	Excepción.	

Tabla 25: Resultados prueba de caja blanca

2.10 Resultados de las pruebas de Estructura

Las pruebas de estructura a pesar de no tener una plantilla para su ejecución, es necesario que se lleven a cabo con la máxima eficiencia posible. Para la realización de la misma se tuvo en cuenta el menú de cada módulo del sistema CCM, verificando que cada enlace del mismo se correspondía con la funcionalidad que en realidad este definía. Luego de haber llevado a cabo todo este proceso antes mencionado solo se encontró un mal direccionamiento en el menú del módulo de catalogación en la funcionalidad de Búsqueda Avanzada. Esto garantiza que el producto CCM cuenta con una buena calidad en cuanto a la estructura de su sistema. En el gráfico que se muestra a continuación se representa de una forma más clara el nivel de error con el que cuenta el sistema CCM después de finalizar estas pruebas.

2.11 Comparación entre resultados reales y esperados.

Haciendo una breve comparación con los errores que se esperaban del sistema y los que se obtuvieron, se puede decir que el sistema a pesar de presentar varios problemas, el mismo realiza la mayor parte de las funcionalidades críticas que fueron definidas en etapas iniciales del producto. Solo contiene algunas fallas de validación, tratamiento de errores y algunos componentes que no están implementados aún, pudiéndose mencionar digitalizar y capturar

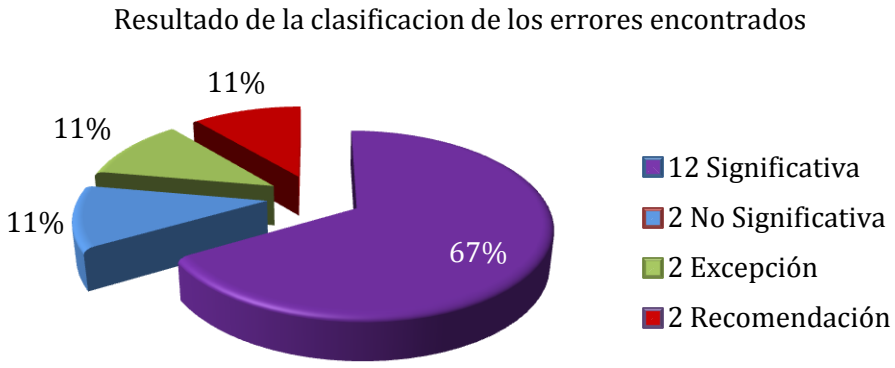
video, ambas del módulo de video. Estas funcionalidades no están implementadas pues todavía no se cuanta con las tarjetas de video que necesita el sistema. A continuación se realizará una pequeña comparación entre los principales defectos encontrados durante todo el proceso de pruebas con los resultados que se esperaban obtener del sistema CCM.

Se esperaba	Se obtuvo	Clasificación
Gestionar todas las planificaciones realizadas por el módulo de administración	No permite insertar, modificar ni eliminar planificaciones.	Significativa.
Gestionar servidor de medias en el módulo de administración.	No permite realizar esta operación de gestión como insertar, eliminar.	Significativa.
Liberar espacio en el servidor de medias	No implementada aun.	Significativa.
Que el caso de uso buscar media aceptara todos los criterios de búsqueda.	Criterios de búsqueda mal, y direccionamiento en el menú incorrecto.	No Significativa.
Que se generaran los reportes correspondientes al módulo de administración	Existen problemas con las variables de entrada en la validación	Significativa.
Que todos los campos de la aplicación se correspondan con los descritos en el modelo de análisis.	No se corresponden.	Recomendación.
Que se exportaran las medias seleccionadas en el módulo de catalogación	En el proceso de exportación si se cancela se cierra la aplicación.	Significativa.
Que se reprodujera las medias seleccionadas en el módulo de catalogación	Cuando se reproduce una media, si se realiza otra operación se cierra la aplicación	Significativa.
Que todos los problemas de conexión con la base de datos fueran consultados con el usuario.	No se informa de forma general al usuario si ocurren problemas de conexión con la BD.	Excepción.
Que todas las validaciones de campos fueran previstas con anterioridad.	Existen muchos campos que no se validan.	Excepción.
Todas las ventanas alineadas.	Ventanas en esquinas de la pantalla (autenticar usuarios).	Recomendación.
Que se reprodujera el archivo de audio	Se reproduce el archivo de audio	No Significativa.

seleccionado.	pero no funciona el pause del mismo. (Por teclado si funciona)	
Subir material digital al servidor con el formato correcto.	No se sube ningún material al servidor (No está implementada esta funcionalidad).	Significativa.
Realizar captura de video.	No se realiza captura de video (No se cuenta con la tarjeta necesaria para realizar esta funcionalidad).	Significativa.
Que se pudiera Digitalizar un video.	No se pude Digitalizar un video No se cuenta con la tarjeta necesaria para realizar esta funcionalidad).	Significativa.
Que el sistema dejara de capturar audio al fichero creado una vez que se presionara la opción pause.	No deja de capturar el audio.	Significativa.
Que el sistema continuara la captura del audio al fichero creado una vez presionado la opción continuar.	No continua la captura.	Significativa.
Capturar audio desde línea.	No se captura.	Significativa.
Que se terminara la grabación exitosamente creando el fichero MP3 antes de que este fuese subido al servidor de medias.	No se crea el fichero MP3.	Significativa.

Tabla 26: Comparación entre resultados esperados y obtenidos

Luego de haber visto los principales defectos encontrados durante la fase de pruebas que se le realizó al sistema CCM, se decide observar estos resultados en un gráfico facilitando con esto que se tenga una mejor vista de los problemas encontrados. A continuación se muestra dicho gráfico.



Como muestra el gráfico anterior queda claro que a pesar de que el producto CCM cumple con las principales funcionalidades críticas que requiere el mismo. Este cuenta con una serie de problemas que atentan contra la calidad del producto final. Por lo que se hace indispensable que todos estos defectos encontrados se resuelvan el menor tiempo posible por el equipo de desarrollo. A continuación de plasman los resultados generales de este trabajo.

2.12 Resultados Generales

En el proceso de ejecución de las pruebas, de modo general, se encontraron errores en cada uno de los casos de usos del producto CCM que fueron sometidos a prueba. Dentro de los errores que se han detectado en el sistema, existen muchos errores que se pueden catalogar de significativos. Siendo el caso de algunas ocasiones no muestran los mensajes de confirmación como en casos de eliminar, hay algunos botones que es necesario verificar porque la implementación es incompleta como son terminar transcodificación del módulo de audio. No se muestran los mensajes de interrupción ya sea de red o con la BD en muchas de las funcionalidades del sistema, existen problemas a la hora de la descripción de los casos de uso, pues existen funcionalidades que no están descritas previamente, o que están descritas y no implementadas. Aunque los errores más graves ocurren en los casos de uso del módulo de catalogación, ejemplo de estos son los CU: exportar media y reproducir media, cabe señalar que una vez que se ejecute esta funcionalidad, si se desea ejecutar otra se cierra la aplicación sin un previo aviso al usuario.

Los errores fueron encontrados durante el funcionamiento de la interfaz del sistema en general, se puede decir que existen algunos aunque no todos de relevante importancia. La carencia de una ayuda del sistema que pueda guiar al usuario en su utilización es uno de ellos, además no

se cuenta con mensajes lo suficientemente explicativos que puedan corregir y guiar al usuario a la hora de utilizar el sistema.

2.13 Conclusiones

Al finalizar este capítulo, se puede decir que el proceso de pruebas realizado al producto CCM se llevó a cabo de forma satisfactoria, pues mediante los casos de prueba confeccionados se obtuvieron muchos de los errores con los que contaba este sistema. En este capítulo quedaron documentados todos los defectos encontrados, luego de haber realizado las pruebas de función (caja negra (CN) y caja blanca (CB)), integración (CN y CB) y estructura, con el fin de que fueran corregidos por el equipo de desarrollo en el menor tiempo posible. Se realizó una comparación entre los resultados que se esperaban obtener de producto CCM y los resultados que se obtuvieron realmente, teniendo con esto una visión más clara acerca de la calidad que presenta el sistema CCM en la actualidad. En fin luego de haber culminado este capítulo se cumplió el objetivo fundamental del mismo, que era dar a conocer todos los defectos con los que contaba el producto luego de aplicar las pruebas correspondientes.

Conclusiones Generales

En toda la labor realizada en este trabajo de diploma al diseñar y aplicar pruebas al software CCM se obtuvieron los siguientes resultados:

1 El aseguramiento de la calidad del software permite reducir notablemente los costos de producción e implantación, y proporciona mayor confianza en el cumplimiento de los requisitos del cliente, siendo las pruebas de software un elemento imprescindible para asegurar la calidad y para verificar el cumplimiento de los requisitos funcionales y los impuestos por el cliente.

3 Se enfatizó en la metodología de desarrollo del software definida por el proyecto CCM específicamente en la fase de pruebas. Logrando con esto tener una visión más profunda en cuanto a las actividades, trabajadores y artefactos involucrados en el mismo.

4 La confección del plan de pruebas para el sistema CCM permitió describir la estrategia, recursos, planificación de las pruebas y el cronograma de pruebas para el producto CCM garantizando con esto que todo el proceso de pruebas fuera de una forma más clara y tangible.

5 Se diseñaron y aplicaron 22 casos de pruebas de caja negra al producto CCM logrando con esto recolectar una cifra de 31 errores, divididos en 4 grupos principales como son: Significativos, No Significativos, Excepciones y Recomendaciones.

6 Se diseñaron y aplicaron 8 casos de pruebas de caja blanca al producto CCM logrando con esto recolectar una cifra de 6 errores.

7 Se documentaron los resultados correspondientes de las pruebas de estructura al sistema CCM.

8 Se documentaron todos los defectos encontrados de las pruebas de caja negra en los documentos de No conformidades correspondiente a cada módulo por separado. Garantizando con esto que los mismo fuesen corregidos por el equipo de desarrollo en el menor tiempo posible.

9 De los 24 casos de usos que fueron probados se detectaron un total de 41 errores permitieron obtener el siguiente resultado: 51.2 % de los errores encontrados están clasificados en significativos, 4.87 % de recomendación, 14.6 % de no significativos, 29, 2 % de Excepción.

10 Se hicieron las recomendaciones necesarias, con el fin de que el producto CCM al ser desplegado funcionase con la calidad que espera el cliente.

Recomendaciones

Una vez concluido este trabajo y dada la importancia que tiene el proceso de pruebas para cualquier sistema informático, se recomiendan los siguientes aspectos que se mencionan a continuación.

1. Hacer extensiva la fase de prueba a los demás proyectos de la facultad, llevando la documentación necesaria cada vez que se le apliquen.
2. Realizar un estudio más profundo acerca de la calidad de software, de su importancia en el proceso de desarrollo y acerca de la importancia de las pruebas de software como parte del aseguramiento de la calidad y las últimas tendencias internacionales en esta materia.
3. Que se le apliquen las pruebas a los caso de usos que no fueron sometidos a prueba completando en su totalidad los CU de proyecto CCM y a los que fueron sometidos a prueba que se le corrijan todos los errores encontrados.
4. Perfeccionar el plan de pruebas teniendo en cuenta la retroalimentación obtenida con la aplicación de las pruebas a los módulos del proyecto CCM.
5. Ya que el producto CCM es un sistema que trabaja con medias muy pesadas[10] se recomienda que se le apliquen pruebas de volumen para verificar el tráfico con la base de datos.
6. Que se le aplique pruebas de tensión y seguridad, una vez que el producto ya este desplegado, para verificar cómo funciona el mismo antes situaciones anormales y que los datos solo sean accedidos por las personas que cuentan con los privilegios necesarios.

[10] *Término que se utiliza en la informática para clasificar el tamaño de un archivo, este puede clasificarse en megabyte, gigabyte, terabyte.*

REFERENCIAS BIBLIOGRÁFICAS CITADAS

1. Eumed. [En línea] [Citado el: 23 de noviembre de 2009.] http://www.eumed.net/libros/2009c/584/Descripcion%20de%20las%20metodologias%20existentes%20para%20el%20desarrollo%20de%20software.htm.

2. **Mc-Graw-Hill.** *Pressman, R.Ingenieria del software: un enfoque práctico.* 2002.

3. sensagen. *sensagen.* [En línea] [Citado el: 20 de Enero de 2010.] http://diccionario.sensagent.com/asegurar/es-es/.

4. rincondelvago. *rincondelvago.* [En línea] [Citado el: 19 de 2 de 2010.] http://html.rincondelvago.com/aseguramiento-de-calidad.htm.

5. SOYUZEXPERTIZA. *SOYUZEXPERTIZA.* [En línea] [Citado el: 21 de 2 de 2010.] http://www.gost-soex.ru/es/DICCIONARIO-DE-TERMINOS-DE-CERTIFICACION.shtml.

6. **McCall.J.** Factors in software Quality:General Electric. *Factors in software Quality:General Electric.* 1977.

7. **McGraw-Hill.** *R.S. Pressman.Ingeniería de Software. Un enfoque Práctico.* s.l. : 4ta Edición, 1998.

8. **Lic. Carlos Galindo González, y Dr. Ramiro Pérez Vázquez.** Rational Unified Process® . *Rational Unified Process® .* 2009.

9. **Ing: Rondón, Yoandri Quintana.** Desarrollo de la arquitectura del proyecto Captura y Catalogación de Medias. [aut. libro] Yoandri Quintana Rondón. *Desarrollo de la arquitectura del proyecto Captura y Catalogación de Medias.* C.Habana : s.n., 2009.

10. **calisoft.** Curso calidad. *Introducción a la Pruebas de software.* C. Habana : s.n., 2009.

11. **Wiley, John.** *Software Engineering Standards.* s.l. : IEEE Press, 2001. ANSI/IEEE Standard 829-1983 .

12. **Hernandez, Yanio.** Documento Visión. Proyecto CCM 1.0. 2009. 1.0.

13. **Hernandez Heredia, Ing: Yanio.** Documento Visión. Proyecto CCM 1.0. 2009. 1.0.

14. PROMONEGOCIOS.NET. *PROMONEGOCIOS.NET.* [En línea] [Citado el: 3 de Marzo de 2010.] http://www.promonegocios.net/proyecto/evaluacion-proyectos.html.

15. *ESTIMACIÓN DEL ESFUERZO BASADA EN CASOS DE USO.* **Peralta, Mario.** Centro de Ingeniería del Software e Ingeniería del Conocimiento (CAPIS) Escuela de Postgrado. Instituto Tecnológico de Buenos Aires : s.n.

16. **Hernandez Heredia, Ing: Yanio.** *Especificación de requisitos del software.* C.Habana : s.n., 2009.

17. **Alonso, Ing: Zorilín.** *Especificación de requisitos del software.* Ciudad Habana : s.n., 2008.

18. **Guardia, Ing: Maria Dolores.** *Especificación de requisitos del software.* Ciudad Habana : s.n., 2008.

19. **Ing: Zorilin Alonso Guerrero, Aneli Valdés.** *Modelo de Sistema. Captura y Catalogación de Medias (CCM).* C.Habana : s.n., 2009.

20. **Heredia, Ing: Yanio Hernandez.** *Modelo de Sistema. Captura y Catalogación de Medias (CCM).* C.Habana : s.n., 2009.

21. **Suárez Pérez, Ing: Jean Michel.** *Modelo de Sistema. Captura y Catalogación de Medias (CCM).* C.Habana : s.n., 2009.

22. **Macias, Maria de Dolores Guardia.** *Modelo de Sistema. Captura y Catalogación de Medias (CCM).* C.Habana : s.n., 2009.

23. Microsoft. *Microsoft.* [En línea] [Citado el: 10 de Abril de 2010.] http://www.microsoft.com/latam/protect/yourself/password/create.mspx.

24. Mitecnologia. [En línea] [Citado el: 15 de noviembre de 2009.] http://www.mitecnologico.com/Main/CalidadDelSoftware.

25. **Mary Belh Chrissis, Mike Konrad, Sandy Shrum.** *CMMI for Development v12.* 2006.

26. *Investigación sobre estado del arte en diseño y aplicación de pruebas de software.* **Johanna Rijas Rojas, Emilio José Barrios.** 2007.

27. Sabueso Web. [En línea] [Citado el: 5 de 12 de 2009.] http://sabuesoweb.com/blog/2007/07/18/metodos-de-caja-blanca-y-caja-negra/.

28. Universidad Distrital Francisco Jose de Caldas. [En línea] [Citado el: 6 de 12 de 2009.] http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

29. Informática Profesional. [En línea] [Citado el: 10 de 12 de 2009.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jmeter.

30. Especialista Universitario Java Enterpise. [En línea] 2009-2010. [Citado el: 11 de diciembre de 2009.] http://www.jtech.ua.es/tutoriales/apuntes/sesion-junit-apuntes.htm .

31. Collan blog. [En línea] [Citado el: 6 de diciembre de 2009.] http://javier.callon.org/complejidad-ciclomatica.

32. TecnoRetales. [En línea] TecnoRetales is proudly using the Buddy theme. Powered by WordPress, 2009. [Citado el: 11 de diciembre de 2009.] http://www.tecnoretales.com/programacion/phpunit-tests-unitarios-en-php/.

33. Departemanto de lenguajes y sistemas Informáticos:Universidad de Granadas. [En línea] [Citado el: 2 de Diciembre de 2009.] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/prod_des/documento/plan_pruebas_d.php.

34. eleZeta:la odisea de ser aprendiz. [En línea] WordPress. [Citado el: 3 de Noviembre de 2009.] http://elezeta.net/2004/08/27/extreme-programming-xp/.

35. Epidataconsulting. [En línea] 2009. [Citado el: 25 de noviembre de 2009.] http://www.epidataconsulting.com/tikiwiki/tiki-index.php?page=Scrum.

36. **Oberto Canales Mora.** Adictos al Trabajo. [En línea] 2007. [Citado el: 10 de 1 de 2100.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=cmmi.

37. **Collazo, Manuel.** *Técnicas de prueba del software. Estrategias de prueba del software.* s.l. : IEEE Standard Glossary of Software Engineering Terminology 1990, 2003.

38. LCD. *Laboratorio Docente de Computación.* [En línea] 2008. [Citado el: 22 de 11 de 2009.] http://www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html.

REFERENCIAS BIBLIOGRÁFICAS CONSULTADAS

1. **McGraw-Hill.** R.S. Pressman.Ingeniería de Software. Un enfoque Práctico. s.l. : 4ta Edición, 1998.

2. Lic. Carlos Galindo González, y Dr. Ramiro Pérez Vázquez. Rational Unified Process® . Rational Unified Process® . 2009.

3. Universidad Distrital Francisco José de Caldas [En línea] 2009 [Citado el: 11 de 11] *2009]http://www.udistrital.edu.co/comunidad/grupos/arquisoft/index.php?id=81&type=1.*

4. Grupo soluciones GSInnova [En línea] 2009 [Citado el : 25 de 10 de 2009.] *http://www.rational.com.ar/herramientas/rup.html*

5. submit articles directory [En línea] 2005-2010 [Citado el 6 de 3 de 2010.] *http://e-articles.info/t/i/1660/l/es/.*

6. Informatizate María A. Mendoza Sánchez [En línea] 2008 [Consultado el: 2 de 11 2009]*http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.*

7. Compute-rs.com [En línea] 2008 [Consultado el: 3 de 12 del 2009] *http://www.compute-rs.com/es/consejos-2066683.htm.*

8. TecnoRetales. [En línea] TecnoRetales is proudly using the Buddy theme. Powered by WordPress, 2009. [Citado el: 11 de diciembre de 2009.] http://www.tecnoretales.com/programacion/phpunit-tests-unitarios-en-php/.

9. Departemanto de lenguajes y sistemas Informáticos:Universidad de Granadas. [En línea] [Citado el: 2 de Diciembre de 2009.] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/prod_des/documento/plan_pruebas_d.php.

10. eleZeta:la odisea de ser aprendiz. [En línea] WordPress. [Citado el: 3 de Noviembre de 2009.] http://elezeta.net/2004/08/27/extreme-programming-xp/.

11. Epidataconsulting. [En línea] 2009. [Citado el: 25 de noviembre de 2009.] http://www.epidataconsulting.com/tikiwiki/tiki-index.php?page=Scrum.

12. **Oberto Canales Mora.** Adictos al Trabajo. [En línea] 2007. [Citado el: 10 de 1 de 2100.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=cmmi.

13. **Collazo, Manuel.** *Técnicas de prueba del software. Estrategias de prueba del software.* s.l. : IEEE Standard Glossary of Software Engineering Terminology 1990, 2003.

14. LCD. *Laboratorio Docente de Computación.* [En línea] 2008. [Citado el: 22 de 11 de 2009.] http://www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html.