UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

COMPONENTE PARA EL MONITOREO DE SEÑALES DIGITALES AUDIOVISUALES

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTOR: DILENIS CEPERO OLIVA

TUTOR: ING. EKATERINA ALEJOVNA RAMÍREZ

La Habana, 2011

"Año 53 de la Revolución"

"El aspecto fundamental en el cual la juventud debe señalar el camino es precisamente en el aspecto de ser vanguardia en cada uno de los trabajos que le compete."

DEDICATORIA

A mis padres, a mi hermana y a mi sobrina por el amor, la confianza, el apoyo y la dedicación que me han brindado siempre, por ser mis guías y mi fuerza. Los quiero mucho.

AGRADECIMIENTOS

A mi mamá por todo el amor y la confianza que siempre me ha dado, por toda su dedicación, esfuerzo, sacrificio y apoyo, por estar siempre a mi lado venciendo todos los obstáculos, por ser la mejor madre del mundo.

A mi papá por confiar siempre en mí, por su amor y cariño, por todo su esfuerzo y apoyo brindado, por estar siempre a mi lado, por ser el mejor padre del mundo.

A mi hermana por su apoyo incondicional, por todo el amor y cariño que siempre me ha dado, por estar siempre cuando la necesito, por ser la mejor hermana del mundo.

A mi bella sobrina por alegrarme siempre con sus travesuras y hacerme olvidar todas las dificultades, por ese gran amor y cariño que desprende su pequeño corazoncito.

A mi tías, Teresa, Carmita, Marisela y a mis tíos, Pucho y Miguel por su apoyo incondicional y por la preocupación constante.

A toda mi maravillosa familia en general, por preocuparse siempre por mi bienestar y poyarme en todo.

A mi amiga Yanisley por todo su apoyo en estos 5 años, por toda la confianza que siempre me ha brindado, por estar presente siempre que la necesito y por aguantarme todas las malcriadeces, por ser otra hermana para mí.

A mi amigo Pupo por todo el apoyo que me ha dado cada vez que lo necesito, porque siempre puedo contar con él.

A mis amigas Yudith, las Susys, Darma, Odalys, Yaimit por todos los momentos malos y buenos que pasamos juntas.

A todos los compañeros de los cursos anteriores, por todos los momentos maravillosos que juntos compartimos.

A mi tutora por todo el tiempo dedicado.

A la UCI por permitirme conocer a todas las personas maravillosas con que conviví y que no voy a olvidar.

A la Revolución, por permitirme realizar este sueño.

A todas las personas que contribuyeron con la realización de este trabajo.

Muchas gracias a todos.

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora 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 _____.

Dilenis Cepero Oliva

_____

Ing. Ekaterina Alejovna Ramírez

_____

DATOS DE CONTACTO

Ing. Ekaterina A. Ramírez Ramírez: Graduada de Ingeniera en Ciencias Informáticas por la Universidad de las Ciencias Informáticas, en el año 2009. Trabaja en la Facultad 6 de la Universidad de las Ciencias Informáticas. Ha recibido varios cursos sobre procesamiento de imágenes digitales. Imparte en la universidad el curso de pregrado "Administración de Linux". Sus intereses de investigación son el procesamiento de imágenes digitales (en general) y la administración de bases de datos (PostgreSQL). Correo electrónico: earamirez@uci.cu.

RESUMEN

El presente trabajo describe el desarrollo de un componente para el monitoreo de señales digitales audiovisuales, con el objetivo de lograr uniformidad en la detección y tratamiento de eventos en señales digitales audiovisuales en el departamento de Señales Digitales de la Universidad de la Ciencias Informáticas. El componente será capaz de visualizar las medias deseadas y detectar si ocurre, algún error en la transmisión de las mismas.

A lo largo de la investigación se estudiaron diferentes soluciones para monitorear las señales digitales audiovisuales que facilitaron su comprensión y elaboración. Además, para la implementación de la aplicación fueron utilizadas varias herramientas y tecnologías como son: el sistema gestor de base de datos PostgreSQL, la librería LibVLC, el framework Qt, el lenguaje de programación C++ y la herramienta Visual Paradigm.

Con la instalación y puesta en práctica del componente para el monitoreo de señales digitales audiovisuales, se facilitará la detección y tratamiento de eventos en señales digitales audiovisuales y se eliminarán los esfuerzos de desarrollos en funcionalidades duplicadas en el departamento de Señales Digitales de la Universidad de la Ciencias Informáticas.

PALABRAS CLAVES: audiovisuales, componente, monitoreo.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

A lo largo de la historia, las revoluciones tecnológicas e industriales han constituido hechos trascendentales en la sociedad. En el siglo XIX, la Revolución Industrial marcó el devenir del hombre como entidad social y cultural. Posteriormente la llegada de la era digital y, con ella, las nuevas tecnologías, generó un crecimiento tecnológico sin precedentes, que no sólo ha favorecido una mejora en la calidad de los servicios, sino un aumento significativo en la diversidad de los mismos.

Las técnicas establecidas por esta nueva etapa digital constituyen un conjunto de tecnologías cuyas aplicaciones despliegan un amplio abanico de posibilidades a la comunicación humana. Dentro de estas oportunidades florece la nueva revolución del audiovisual, ya no se está ante el viejo audiovisual digitalizado sino que realmente ha nacido un nuevo concepto de audiovisual, con su lenguaje, su industria, sus profesionales y sus espectadores.

El origen de los audiovisuales viene dado por la introducción de los equipos de grabación de imágenes y sonidos en la televisión y radio, y después, en un segundo período, por el inicio de las redes telemáticas como el videotexto, de los juegos de vídeo, del ordenador portátil y finalmente, de Internet y la telefonía móvil. (Castañeda, 2006)

A finales del siglo XX, cuando se generó el desarrollo y el ingreso de las nuevas redes digitales de banda ancha los audiovisuales retomaron gran importancia, debido a que los considerables volúmenes de información que contenían, se volvieron en un indiscutible reto tecnológico de todas aquellas personas o entidades responsables de su distribución o transmisión a una gran velocidad, con una óptima facilidad de manejo, tratamiento y acceso para sus usuarios.

Anteriormente los audiovisuales habían tenido un alto valor comercial y cultural principalmente en el campo del ocio y el entretenimiento, pero con la nueva era digital se convirtieron en extraordinarias fuentes de contenidos de las redes digitales que podían ofrecer de forma inmediata recursos especializados en tareas generales o específicas tanto de interés cultural como para la divulgación o adquisición de nuevos y antiguos conocimientos.

En los últimos años, la información y documentación se incrementa cotidianamente, los archivos que contienen los audiovisuales no dejan de crecer, se multiplican y su utilización se vuelve por momentos muy intensa e influye en los procedimientos de almacenamiento, conservación, catalogación y acceso a

los mismos. Muchos de estos archivos, son indispensables para la creación de nuevos contenidos, y muy valiosos para todas las personas que los utilizan, porque son pruebas o testimonios visuales y/o sonoros de los conocimientos adquiridos hasta el día de hoy.

A raíz del surgimiento y evolución de Internet, se puede visualizar la enorme importancia que tienen los audiovisuales y especialmente los educativos y culturales ya que se puede acceder tanto a un abanico de filmes de ficción, como a clips musicales, noticieros televisivos o cualquier otro programa.

En el mundo entero existen un gran número de instituciones y centros que utilizan las señales digitales audiovisuales para satisfacer sus necesidades y nuestro país no se queda atrás. Actualmente Cuba posee diversas escuelas e instituciones donde se utilizan estos medios para el entretenimiento, el aprendizaje y la culturización de la sociedad, por ejemplo la Universidad de las Ciencias Informáticas surgida para irrumpir en el desarrollo de software e informatizar la sociedad cubana. Dentro de las facultades con que cuenta esta universidad se encuentra la facultad número 6 que es la encargada de dar soluciones informáticas en el campo del procesamiento de Señales Digitales y Geoinformática del Centro de Desarrollo GEySED. Cada proyecto de este centro, individualmente posee un sistema para monitorear las señales audiovisuales digitales, pero en algunos casos presentan labores muy parecidas incluso existen iguales, por ejemplo visualizar en pantalla completa o de manera independiente la señal son requisitos tanto del proyecto Video Vigilancia como del proyecto Plataforma de Transmisión Abierta de Radio y Televisión (PTARTV), sin embargo, esta funcionalidad es igual para ambos proyectos, además existen otras que son diferentes por pocos elementos, por ejemplo visualizar datos tiene pocas características que varían de un proyecto a otro, son funcionalidades que se pueden realizar de forma genérica, por lo que es necesario encontrar una solución que agrupe en una sola las exigencias iguales e incorpore los requisitos particulares de cada proyecto minimizando de esta manera los esfuerzos de desarrollos en funcionalidades duplicadas.

Teniendo en cuenta lo mencionado anteriormente se define como **problema a resolver** ¿Cómo lograr uniformidad en la detección y tratamiento de eventos en señales digitales audiovisuales en el departamento de Señales Digitales?

Determinando como **objetivo general** de la investigación: Desarrollar un componente informático que permita el monitoreo de señales digitales audiovisuales para lograr uniformidad en el departamento de Señales Digitales.

Para darle solución a este problema se define como **objeto de estudio** los procesos de monitoreo de señales digitales audiovisuales.

El resultado de la investigación tendrá como **campo de acción** la informatización de los procesos de monitoreo de señales digitales audiovisuales en el departamento de Señales Digitales.

Por la tanto la **idea a defender** sería: Si se dispone de un componente informático que logre uniformidad en la detección y tratamiento de eventos en señales digitales audiovisuales en el departamento de Señales Digitales entonces se eliminarán los esfuerzos de desarrollos en funcionalidades duplicadas en el mismo.

Durante el desarrollo de esta investigación, y para dar respuesta al problema científico planteado, se aplicarán los siguientes **métodos científicos**:

Métodos teóricos:

- Histórico-Lógico: para lograr desarrollar un estudio y valoración de la evolución que ha tenido el proceso de monitoreo de señales audiovisuales digitales en diferentes momentos históricos.
- Analítico-Sintético: para elaborar una síntesis de los procesos de monitoreo de señales audiovisuales digitales, de su funcionamiento, los rasgos que lo caracterizan y distinguen a partir del análisis de las diferentes teorías.
- Modelación: para una mejor comprensión del proceso, para descubrir nuevas relaciones y cualidades de los procesos de monitoreo de señales digitales audiovisuales.

Como **posibles resultados** de la investigación se tendrá:

Componente de software para la detección y el tratamiento de los eventos en señales digitales audiovisuales.

Las **tareas de la investigación** que se proponen para lograr la realización de la misma son:

- Definir el estado del arte a partir de la investigación de herramientas de monitoreo de señales digitales audiovisuales existentes.
- Describir las herramientas y metodologías seleccionadas para el desarrollo del componente.
- Identificar las características del entorno de trabajo y del futuro sistema.

    - Realizar el modelo de dominio.
    - Realizar el levantamiento de requisitos funcionales y no funcionales.
- Diseñar el componente para el monitoreo de señales digitales audiovisuales
- Implementar el componente para el monitoreo de señales digitales audiovisuales.
- Realizar las pruebas al componente.

Este trabajo tendrá la siguiente estructura:

En el Capítulo 1 “Fundamentación teórica”, se hace referencia al estado del arte del objeto de estudio de la presente investigación, se enuncian conceptos que posibilitan un mejor entendimiento de lo planteado en la situación problemática y el marco del problema en sentido general. Además, se realiza un análisis de las tecnologías, metodologías y herramientas actuales, quedando seleccionadas las que se va a utilizar para el desarrollo del componente.

En el Capítulo 2 “Presentación de la solución propuesta”, se presenta la solución propuesta a partir de la realización de un estudio inicial del entorno que encierra el problema a resolver con la misma. Se ofrece el modelo de dominio, se identifican los requerimientos funcionales y no funcionales, y se describe el modelo de casos de usos del componente.

En el Capítulo 3 “Diseño de la solución propuesta”, Se efectúa el diseño completo de la aplicación, punto de partida para el desarrollo de la implementación de la misma, tratando aspectos fundamentales como los diagramas de interacción y los diagramas de clases.

En el Capítulo 4 “Implementación y proceso de pruebas”, se plasmarán las generalidades del Flujo de Trabajo de Implementación así como los resultados de las pruebas realizadas al componente.

Finalmente, se presentan las Conclusiones, Recomendaciones, Bibliografía y Anexos.

CAPÍTULO 1: Fundamentación Teórica del componente para el monitoreo de señales digitales audiovisuales

1.1 Introducción

En este capítulo se abordan los aspectos investigativos para la realización del trabajo. Se realizará un análisis sobre el estado en que se encuentran en el mundo los sistemas similares a la aplicación a desarrollar y de las tecnologías adecuadas para llevar a cabo el sistema.

1.2 Conceptos asociados al dominio del problema

1.2.1 Audiovisual

El término del arte audiovisual empieza a usarse en Estado Unidos en los años 1930 con la aparición del cine sonoro. Sin embargo, empieza a teorizarse en Francia durante la década de los años 1950 para referirse a las técnicas de difusión simultáneas. Es a partir de entonces cuando el concepto se amplía y el término se sustantiva. En el terreno de los medios de comunicación de masas, se habla de lenguaje audiovisual y comunicación audiovisual.

Audiovisual es todo aquello que se refiere a las imágenes en movimiento y/o a los sonidos grabados, registrados en película, cinta magnética, disco o cualquier otro medio actualmente conocido o por inventar.(Castañeda, 2006)

Audiovisual hace referencia a la integración e interrelación plena entre lo auditivo y lo visual. De esta forma la percepción se realiza de forma simultánea, se crean nuevas realidades sensoriales, a cada sonido le corresponde una imagen, lo que no aporta uno lo aporta el otro.(Martínez Alvarez, 2008)

Teniendo en cuenta lo planteado anteriormente se puede deducir que el término *audiovisual* se refiere a la combinación de lo visual y lo auditivo, a todas las imágenes acompañadas de sonidos.

1.2.2 Monitoreo

Monitoreo es un proceso de medición sistemática de indicadores objetivamente verificables de un proyecto con el fin de determinar el grado de consecución de los objetivos previstos. (Hernández Becerra, 1993)

El *monitoreo* es el seguimiento sistemático y periódico de la ejecución de una actividad o proyecto para verificar el avance en la ejecución de la Meta Física (eficacia), la adecuada utilización de recursos para lograr dicho avance (eficiencia) y la consecución de los objetivos planteados durante el proceso de ejecución (efectividad), con el fin de detectar, oportunamente, deficiencias, obstáculos y/o necesidades de ajuste.(Ministerio de Educación Perú, 2007)

Por lo tanto, *monitoreo* se refiere al seguimiento continuo que se realiza a algún proceso para verificar la eficiencia y eficacia del mismo identificando sus logros y debilidades para detectar las futuras deficiencias.

1.2.3 Señal digital

La *señal digital* es un tipo de señal generada por algún tipo de fenómeno electromagnético en que cada signo que codifica el contenido de la misma puede ser analizado en término de algunas magnitudes que representan valores discretos, en lugar de valores dentro de un cierto rango.(Vargas)

Cuando una señal discreta en el tiempo sólo puede tomar valores de amplitud discretos, entonces se trata de una señal discreta tanto en el tiempo como en amplitud. Este tipo de señales ha cobrado una gran importancia en las comunicaciones digitales, ya que los sistemas modernos de telecomunicaciones son eficientes y efectivos precisamente debido a este tipo de señales. A las señales que son discretas en el tiempo y en amplitud se les denomina *señales digitales*. (ILCE (Instituto Latinoamericano de la Comunicación Educativa), 2006)

Conocido lo anterior se puede concluir que la señal digital es generada por un fenómeno electromagnético y es discreta en el tiempo y en amplitud.

El monitoreo de las señales digitales audiovisuales consiste en recoger la información y los datos necesarios sobre estas señales para sintetizar, analizar y usar la información para tomar decisiones y realizar acciones.

1.3 Análisis de otras soluciones existentes

Actualmente en el mundo existen varias soluciones para monitorear las señales digitales audiovisuales, a continuación se describen algunas de estas soluciones.

1.3.1 ImageRecorder NewGen

ImageRecorder NewGen es un software especialmente diseñado para centros de monitoreo donde es vital aprovechar al máximo el potencial de las cámaras IP y servidores de video marca Axis, así como facilitar la operación en situaciones críticas. Cuenta con funciones de visualización individual por cámara, por secuencia programada de cámara ó mosaico, vistas múltiples en mosaico de hasta 64 cámaras simultáneas en una misma ventana, manejo de mapas, audio en visualización y grabación, manejo de alarmas. Permite almacenar el video en el disco duro de manera continua, programada por días y horarios específicos ó por movimiento analizando la imagen completa ó solo parte de ella, posteriormente puede ser consultado por fecha, hora y cámara. (Sistemas de Seguridad, Vigilancia y Monitoreo Remoto)

Algunas características que presenta esta solución son las siguientes:

- Cuenta con un supervisor de procesos que continuamente verifica el estado de las cámaras, comunicaciones y procesos de grabación el cual reporta por correo electrónico cualquier anomalía.
- Cuenta con funcionalidad para manejo de alarmas disparadas por dispositivos externos conectados a las cámaras.
- Permite Zoom Digital en reproducción de históricos para tener mejor percepción de los hechos.

1.3.2 Tedial

Tedial es una compañía que nace de la investigación universitaria, es el resultado de la unión de experiencias profesionales con amplios conocimientos sobre los últimos avances relacionados con los sistemas de información audiovisual. Presenta varias aplicaciones para el procesamiento de los archivos audiovisuales ya sean sistemas de vigilancia, de gestión y almacenamiento, digitalización, catalogación, archivo entre otros. (Tedial, 2009)
A continuación se describen algunos módulos de esta compañía.

TD MPM es un sistema de gestión de flujos de archivos audiovisuales que permite definir, planificar y monitorizar las transferencias y transformaciones de los audiovisuales entre los distintos sistemas que componen una instalación de televisión. A continuación algunas características de este módulo. (Tedial, 2009)

- Escalabilidad: la capacidad del MPM puede verse incrementada simplemente añadiendo nuevos servidores.
- Configuración: sencilla configuración de usuario gracias al Flow Builder Editor que permite realizar cualquier cambio en cualquier momento de forma temporal o permanente.
- Estructura del formato y restauración: verificación del formato estándar y restauración siempre que sea necesaria.
- Sincronización de la base de datos: MPM intercambia metadatos con bases de datos externas.
- Web: MPM introduce la publicación Web.

1.3.3 ZoneMinder

ZoneMinder es un conjunto de aplicaciones que conjuntamente proporcionan una completa solución de video vigilancia permitiendo capturar, analizar, grabar y monitorizar cualquier cámara de Circuito Cerrado de Televisión (CCTV) conectada a una máquina basada en Linux. Con ZoneMinder podemos manejar nuestro sistema de vigilancia y ver las imágenes de las cámaras de seguridad desde cualquier parte del mundo.(Software Libre, 2010)

Algunas características de esta aplicación son:

- Soporta cámaras de Video, cámaras USB y cámaras IP.
- Posibilidad de definir varias zonas de interés con varios grados de sensibilidad.
- Incluye soporte bidireccional del protocolo de automatización de casa X.10 posibilitando por ejemplo que las luces se enciendan cuando se detecte el movimiento.
- Se administra desde una interfaz web, que permite controlar todas las funciones del programa.
- Notificación de eventos por e-mail incluido imágenes de los acontecimientos.

1.3.4 Stream Multiscreen

Stream MultiScreen es un sistema de hardware y software altamente efectivo para monitorear varias señales de audio y video simultáneamente en tiempo real. Está diseñado para controlar en tiempo real varias señales de audio y video de forma simultánea en una pantalla de computadora. Permite crear múltiples configuraciones de ventanas totalmente personalizadas, colocar indicadores de audio y mostrar otros elementos informativos en el monitor. Además de mostrar la señal, el sistema gestiona los parámetros de la señal de video y audio entrante, y provee alertas y mensajes. (Stream Labs, Televisión computer systems, 2010)

Algunas características que presenta esta solución son:

- Soporte para múltiples monitores.
- Control de parámetros de señal y detección de errores.
- Alta calidad de imagen.
- Interfaz amigable.
- La conexión a LAN de Stream MultiScreen permite unir varios dispositivos en una sola red y controlarlos completamente desde una estación de trabajo.

Los sistemas anteriormente mencionados poseen diversas y eficientes funcionalidades para el monitoreo de las señales digitales audiovisuales, pero se necesita además que estén basados en software libres y no todos cumplen con este requisito, por ejemplo Stream Multiscreen, Tedial, ImageRecorder NewGen se encuentran desarrollados en plataformas propietarias, lo que implica que queden descartados de esta manera para su utilización. Por el contrario de las aplicaciones anteriores ZoneMinder se encuentran desarrollada en plataforma libre pero tampoco satisface completamente las expectativas del sistema a implementar ya que sólo trabaja con señales digitales desde cámaras de Circuito Cerrado de Televisión y los requisitos del sistema requieren además laborar con estas señales desde otras fuentes como por ejemplo señales de televisión, radio entre otras.

1.4 Metodologías de Desarrollo

La Metodología de desarrollo de software es un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. (Entorno Virtual de Aprendizaje, 2010)

Para el desarrollo de la aplicación se tienen las siguientes metodologías de desarrollo como candidatas:

1.4.1 Programación Extrema (XP)

Está centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP está guiada por una rápida programación y se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, en la reutilización de código, en la realización de pruebas a los principales procesos con el objetivo de tratar de obtener los posibles errores futuros. (Beck, 1999)

1.4.2 SCRUM

Metodología desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, es un término utilizado en el juego de rugby que llevado a la ingeniería significa que el equipo se agrupará para llegar con el apoyo de cada uno de los miembros del proyecto a obtener un producto con calidad que es el objetivo fundamental. Define un proceso empírico, iterativo e incremental de desarrollo, está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.(Calderón, y otros, 2007)

1.4.3 Proceso unificado de Desarrollo (RUP)

El Proceso Unificado es un marco de trabajo genérico que puede especificarse para una gran variedad de sistemas de Software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema de software en construcción está formado por componentes de software interconectados a través de interfaces bien definidas.(Jacobson, y otros, 2000)

Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Cubre el ciclo de vida y desarrollo de software. Su meta es asegurar la producción de software de alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible. Constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo.

Beneficios que aporta RUP:

- Permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, el rendimiento, la reutilización, la seguridad y el mantenimiento del software mediante una gestión sistemática de los riesgos.
- Permite la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos, con una agenda y costo predecible.

- Enriquece la productividad en equipo y proporciona prácticas óptimas de software a todos sus miembros.
- Permite llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para todas las actividades críticas.
- Unifica todo el equipo de desarrollo de software y mejora la comunicación al brindar a cada miembro del mismo una base de conocimientos, un lenguaje de modelado y un punto de vista de cómo desarrollar software.
- Optimiza la productividad de cada miembro del equipo al poner al alcance la experiencia derivada de miles de proyectos y muchos líderes de la industria.

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo consta de 4 fases:

- Inicio: El objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.
- Construcción: En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial, o sea, crear el producto.
- Transición: El objetivo es llegar a obtener el reléase del proyecto.

En cada fase se ejecutarán una o varias iteraciones de tamaño variable según el proyecto, y dentro de cada una de ellas se obtendrá una versión del producto entregable al cliente. A continuación se mencionan los flujos de trabajos, los cuales se dividen en Flujos de trabajo de desarrollo y Flujos de trabajo de soporte que requieren las nuevas actividades anteriormente citadas.

Flujos de trabajo de desarrollo:

- Modelado del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Análisis de requisitos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Análisis y diseño: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

- Implementación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- Prueba: Busca los defectos a lo largo del ciclo de vida.

Flujos de trabajo de soporte:

- Gestión del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Gestión de configuración y cambios: Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos y control de versiones.
- Gestión del entorno: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

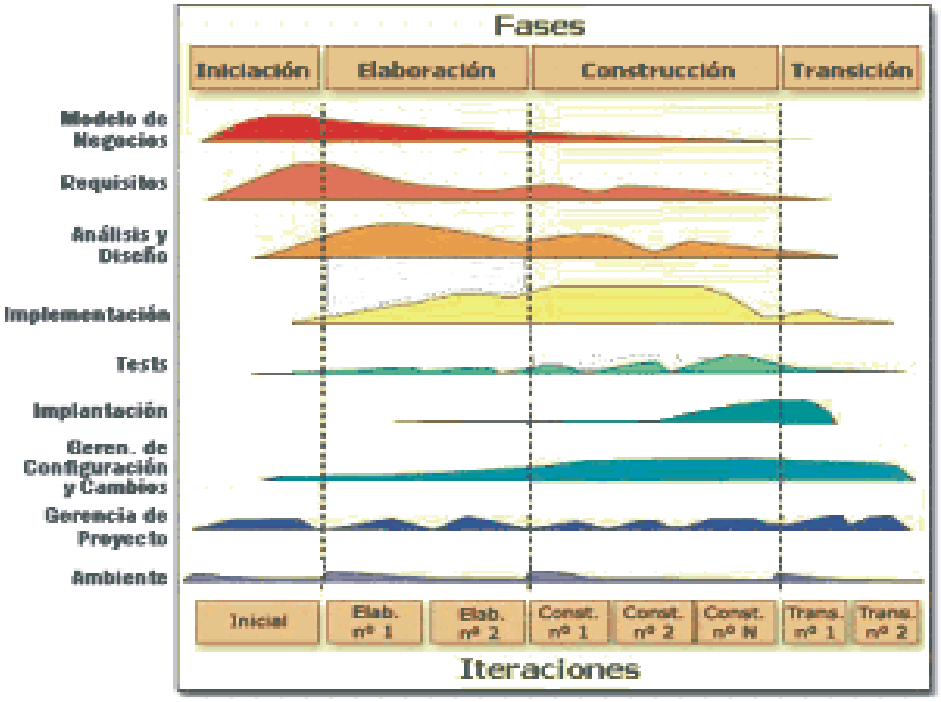

Figura 1 Fases y flujos de trabajos de RUP

1.4.4 ¿Por qué RUP?

Teniendo en cuenta lo descrito anteriormente se descarta el uso de la metodología XP ya que es necesario tener un intercambio constante entre los usuarios y los desarrolladores, aspecto que no es factible para desarrollar el sistema, ya que no se cuenta con un cliente específico, además al igual que Scrum evita la generación documental, ambas se basan más en el desarrollo que en la organización y deben unirse a otras metodologías para completar sus carencias.

Por la importancia que tiene para el componente la elaboración de una eficiente documentación se selecciona RUP debido a que es una metodología bastante orientada al documento, aporta todos los elementos necesarios para desarrollar aplicaciones que requieren de mucha documentación, cubre todo el ciclo de vida y desarrollo de software, no necesita que el cliente forme parte del equipo de desarrollo, la gran cantidad de artefactos que se generan con esta metodología contribuyen a un mejor entendimiento del problema, además es una de las metodologías más generales y más usadas, pues permite adaptarse a cualquier proyecto, es decir, se puede hacer más ágil según se necesite, unifica los mejores elementos de otras metodologías, como es caracterizado por ser iterativo e incremental, permite el refinamiento constante del sistema y la adición de nuevas funcionalidades a cada una de las iteraciones, y como es dirigido por casos de usos y centrado en la arquitectura asegura que el resultado final del producto cumpla con los requisitos planteados por el cliente o usuario.

1.5 Lenguaje Unificado de Modelado (UML)

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas, no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.(Entorno Virtual de Aprendizaje, 2010)

Este lenguaje es fácil de aprender y usar, conforma la colección de las mejores técnicas de ingeniería que han probado ser un éxito en el modelado de sistemas grandes y complejos.

Las ventajas que posibilita la utilización de UML son diversas, por ejemplo el uso de los lenguajes visuales facilita la asimilación y entendimiento del contenido, se minimiza el tiempo invertido en el desarrollo del sistema, la documentación del proyecto se confecciona de una forma ordenada y guiada por casos de uso.

1.6 Herramientas CASE

Case (Ingeniería del Software Asistida por Computadora) comprende un amplio abanico de diferentes tipos de programas que se utilizan para ayudar a las actividades del proceso del software, como el análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas. En la actualidad, todos los métodos vienen con tecnología CASE asociada. Las herramientas CASE también incluyen un generador de código que automáticamente genera código fuente a partir del modelo del sistema y de algunas guías de procesos para los ingenieros de software.(Sommerville, 2005)

Son aplicaciones informáticas que están destinadas a aumentar la productividad en el desarrollo de software y reducir los costos en cuanto a tiempo y dinero.
A continuación se describen para el desarrollo de la aplicación las siguientes herramientas Case como candidatas.

1.6.1 Rational Rose

Rational Rose es una herramienta de diseño orientada a objetos, que da soporte al modelado visual, es decir, que permite representar gráficamente el sistema, permitiendo hacer énfasis en los detalles más importantes, centrándose en los casos de uso y enfocándose hacia un software de mayor calidad, empleando un lenguaje estándar común que facilita la comunicación**.** (Jiménez Lezama, 2005)

Esta herramienta proporciona mecanismos para realizar la Ingeniería Inversa, cubre todo el ciclo de vida de un proyecto: concepción y familiarización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases, brinda la posibilidad de que varias personas trabajen a la vez, permitiendo que cada desarrollador opere en un espacio de trabajo privado.

1.6.2 Visual Paradigm

Visual Paradigm es una herramienta visual de Ingeniería de Software para el modelado, brinda una colección de menús, barras de herramientas y ventanas que forman el área de trabajo, lo cual permite crear diferentes tipos de diagramas en un ambiente completamente visual. Está diseñada para una gran variedad de usuarios, incluyendo Ingenieros de Software, Analistas de Sistema, Analistas de Negocios, por nombrar algunos.(Entorno Virtual de Aprendizaje, 2010)

Esta herramienta ayuda a crear de manera rápida aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Dentro de sus principales características se encuentran: posibilita el uso de varios idiomas, soporta aplicaciones Web, es un producto de calidad, es fácil de instalar, actualizar y posee gran compatibilidad entre ediciones.

Es una herramienta profesional multiplataforma, que proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Posee como peculiaridad sobre el resto de las demás herramientas que cuenta con una potente funcionalidad para la creación de interfaces de usuarios de las aplicaciones.

1.6.3 ¿Por qué Visual Paradigm?

Ambas herramientas presentan grandes potencialidades para el modelado de la aplicación pero se descarta Rational Rose al ser privativa, seleccionando Visual Paradigm ya que posee licencias gratuitas y comerciales, es una herramienta multiplataforma, que permite dibujar todos los tipos de diagramas de clases, presenta amplia facilidad de uso, abundantes tutoriales de UML y una potente funcionalidad para la creación de interfaces.

1.7 Sistemas Gestores de Bases de Datos (SGBD)

El Gestor de Base de Datos es un componente de software encargado de garantizar el correcto, eficiente, íntegro y seguro acceso y almacenamiento de los datos. Además, proporciona una interfaz entre los datos almacenados y los programas de aplicación. Se encarga de:

- Garantizar la privacidad de los datos.
- Garantizar la integridad de los datos, gestionando que los datos que se almacenan en la base de datos satisfagan las restricciones definidas en el esquema de la misma.
- Garantizar la seguridad de los datos realizando procedimientos que puedan recuperar datos después de un fallo que ocasione pérdida o deterioro temporal de los mismos.
- Garantizar el acceso concurrente a la base de datos de forma que varios usuarios puedan acceder al mismo o distinto dato sin que provoque pérdida en la integridad de los datos.

- Interaccionar con el sistema operativo, en concreto con el gestor de archivos.(Nevado Cabello)

Para el desarrollo de la aplicación se tienen los siguientes sistemas gestores de base de datos como candidatos:

1.7.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Incluye características nuevas y mejoradas, que van a aumentar los beneficios en cuanto al diseño de aplicaciones, administración de la base de datos y usuarios. (PostgreSQL, 2010)

Está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales.

A continuación se enumeran las principales características de este gestor de bases de datos:

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.(Pecos, 2002)

1.7.2 Oracle

Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general.(Masip, 2002)

Está considerado como uno de los Sistemas Gestores de Bases de Datos más fuertes y robustos para la elaboración de grandes sistemas donde el principal objetivo sea el manejo fácil y fiable de la información. Tiene implementaciones sobre plataformas libres pero con derechos de licencias, que imposibilitan su implantación en soportes de pocos ingresos monetarios.

A continuación se mencionan algunas de sus características:

- Es una herramienta de administración gráfica que es mucho más intuitiva y cómoda de utilizar.
- Ayuda a analizar datos y efectuar recomendaciones concernientes a mejorar el rendimiento y la eficiencia en el manejo de aquellos datos que se encuentran almacenados.
- Apoya en el diseño y optimización de modelos de datos.
- Apoya en la definición de estándares de diseño y nomenclatura de objetos.
- Documentar y mantener un registro periódico de las mantenciones, actualizaciones de hardware y software, cambios en las aplicaciones y, en general, todos aquellos eventos relacionados con cambios en el entorno de utilización de una base de datos. (Masip, 2002)

1.7.3 MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la licencia GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. Fue creado por la empresa sueca MySQL AB, que mantiene el derecho de autor del código fuente del servidor SQL, así como también de la marca. Aunque es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

MySQL Cuenta con millones de aplicaciones y aparece en el mundo informático como una de las más utilizadas por usuarios del medio. El programa MySQL se usa como servidor a través del cual pueden conectarse múltiples usuarios y utilizarlo al mismo tiempo.

En las últimas versiones se pueden destacar las siguientes características principales:

- El principal objetivo de MySQL es velocidad y robustez.
- Soporta gran cantidad de tipos de datos para las columnas.
- Gran portabilidad entre sistemas, puede trabajar en distintas plataformas y sistemas operativos.
- Cada base de datos cuenta con 3 archivos: Uno de estructura, uno de datos y uno de índice y soporta hasta 32 índices por tabla.
- Aprovecha la potencia de sistemas multiproceso, gracias a su implementación multihilo.
- Flexible sistema de contraseñas y gestión de usuarios, con un muy buen nivel de seguridad en los datos.
- El servidor soporta mensajes de error en distintas lenguas(Cruz Chávez, 2010)

1.7.4 ¿Por qué PostgreSQL?

Se propone el uso de PostgreSQL como gestor de base de datos ya que se encuentra considerado como uno de los gestores de base de datos de código abierto más avanzado del mundo, se caracteriza por su robustez y escalabilidad, además facilita diferentes características que se encontraban solamente en base de datos comerciales de difícil acceso, como por ejemplo Oracle que tiene como mayor inconveniente que es un software privativo y las licencias son extremadamente caras. MySQL también es un potente gestor de software libre pero fue comprado en el 2008 por la empresa norteamericana Sum Microsystems lo que imposibilita que se puedan descargar sus distribuciones.

1.8 Lenguajes de Programación

Las máquinas en general, y las computadoras en particular, necesitan de un lenguaje propio para poder interpretar las instrucciones que se les dan y para que nosotros podamos controlar su comportamiento. Ese lenguaje que permite esta relación con las computadoras es el lenguaje de programación(Lanzillotta, Analía, 2009), el cual está conformado por una serie de reglas sintácticas y semánticas que serán utilizadas por el programador para crear un programa o subprograma a través de ellas. Estos lenguajes se clasifican según el nivel de abstracción en: lenguaje de bajo nivel, que es el código fuente de la máquina, el que la máquina puede interpretar, lenguaje de nivel medio, que es el término entre el lenguaje de la máquina y el lenguaje natural y el lenguaje de alto nivel que es el que está compuesto por elementos del lenguaje natural es decir humano.

En general los lenguajes de programación facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar. Para el desarrollo de la aplicación se tienen los siguientes lenguajes de programación como candidatos:

1.8.1 Java

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. (Lenguajes de Programación, 2010)

Es un lenguaje multiplataforma, diseñado para crear software altamente fiable, para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución, es compatible con los más variados entornos de red, cualquiera sean estos desde Windows 95, Unix a Windows Nt y Mac, para poder trabajar con diferentes sistemas operativos. Tiene la facilidad de cumplir varias funciones al mismo tiempo, gracias a su función de multihilos ya que por cada hilo que el programa tenga se ejecutarán en tiempo real muchas funciones al mismo tiempo.

1.8.2 C#

C# es un lenguaje orientado a objetos creado por Microsoft para su plataforma .NET. Aunque esta plataforma permite desarrollar aplicaciones en otros lenguajes de programación, C# ha sido creado específicamente para .NET, adecuando todas sus estructuras a las características y capacidades de dicha plataforma. Al ser posterior a C++ y Java, los lenguajes orientados a objetos más conocidos hasta entonces, C# combina y mejora gran parte de las características más interesantes de ambos lenguajes.(Cerezo López, y otros, 2007)

Una de sus características es la sencillez ya que elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET por ejemplo no se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple entre otros. Soporta todas las características propias del paradigma de programación orientada a objetos: encapsulamiento, herencia y polimorfismo. Con este lenguaje ahorramos tiempo en la programación ya que tiene una librería de clases muy completa y bien diseñada.

1.8.3 C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma. (Schildt, 1995)

Este lenguaje tiene la posibilidad de redefinir los operadores (sobrecarga de operadores) e identificación de tipos en tiempo de ejecución, permite trabajar tanto a alto como a bajo nivel, con él se pueden programar desde programas muy simples a programas complicados incluyendo sistemas operativos. Sus principales características son las facilidades que brinda para la programación orientada a objetos y para el uso de plantillas o programación genérica.

1.8.4 ¿Por qué C++?

Para la selección del lenguaje se tiene en cuenta los IDEs en sistemas operativos libres. El lenguaje Java es soportado por IDEs de programación libre pero tiene el inconveniente que para poder ejecutar las sentencias se tienen que importar máquinas virtuales lo que conlleva a una compilación muy lenta y se necesita para lograr una mayor eficiencia que la velocidad de procesamiento del componente a desarrollar sea máxima. El lenguaje C# a pesar de ser nativo de la plataforma propietaria .Net se puede utilizar en Linux con el IDE MonoDevelop pero con un rendimiento muy inferior comparado con dicha plataforma y es necesario que el componente goce de un alto rendimiento para monitorear las señales audiovisuales digitales.

Ya que la aplicación a implementar necesita de una gran estabilidad y rendimiento se selecciona el lenguaje C++ porque este combina ambos aspectos, permite un mayor rendimiento en cuanto a consumo de memoria como de trabajo del procesador, posee abundante documentación, cuenta con IDEs en sistemas operativos libres, además está difundido mundialmente y cuenta con una gran comunidad de desarrollo.

1.9 IDE de desarrollo

Los IDEs deben seleccionarse según los lenguajes en que se piensa desarrollar la aplicación, ya que no todos los IDEs soportan estos lenguajes candidatos. Como se ha seleccionado el C++ se tienen como candidatos los siguientes IDEs:
Para el desarrollo de la aplicación se tienen los siguientes IDEs de desarrollo como candidatos:

1.9.1 KDevelop

El proyecto KDevelop fue iniciado en 1998 para diseñar un entorno de desarrollo integrado, fácil de usar, para C/C++ en Unix. Desde entonces está disponible públicamente bajo licencia GPL.(La Zona Linux, 2010)

KDevelop es un IDE para programar en C y C++ en Linux. Integra gran cantidad de herramientas, srcipts (archivo de órdenes o archivo de procesamiento por lotes), y las plantillas en una interfaz de usuario común. Tiene extensibles módulos que pueden ser cargados en tiempo de ejecución y sobre la demanda, lo que le permite a su vez realizar un diagnóstico sólo de aquello que realmente necesita, así como la documentación y optimización de herramientas de ayuda.

Contiene algunas herramientas de edición, distintos navegadores que ayudan al mantenimiento de las clases y sus relaciones con la programación orientada a objetos. Posee asistentes y potentes API (Interfaz de Programación de Aplicaciones) para un desarrollo rápido y como principal ventaja la posibilidad de brindar múltiples funcionalidades reduciendo a su vez el número de líneas de código.

1.9.2 Qt Creator

Qt Creator es un IDE multiplataforma muy completo, creado para desarrollar aplicaciones en C++ de manera sencilla y rápida. Algunas de sus principales características son: posee un avanzado editor de código C++, diseñador de formularios integrado, herramientas para la administración y construcción de proyectos, completado automático, depurador visual.(Geeks & Linux Atelie!, 2010)

Permite construir interfaces de usuario complejas de una forma visual y rápida ya que incluye un editor de texto con autocompletado, diseñador de interfaces gráficas, gestión de proyectos, sistema de depuración e integración con sistemas de control de versiones.

Qt Creator se integra bien con el C++, facilita el trabajo no sólo a los desarrolladores que están acostumbrados a utilizar el ratón, sino también a los desarrolladores que se sienten más cómodos con el teclado ya que posee una amplia gama de métodos abreviados de teclado y navegación disponibles para ayudar a acelerar el proceso de desarrollo de una aplicación.

1.9.3 ¿Por qué Qt Creator?

Para el desarrollo de la aplicación se selecciona el IDE Qt Creator, ya que cuenta con una interfaz agradable y con el QtDesigner que ayuda a diseñar formas de interfaz de usuario de forma cómoda. Presenta una herramienta de búsqueda eficaz para buscar fácilmente las clases, métodos y archivos y un poderoso editor de código con múltiples funcionalidades. Integra todas las herramientas de Qt de forma que hace mucho más fácil la programación y compilación del código. Tiene como ventaja que es soportado por la misma compañía que soporta a Qt, por lo que propicia una mayor integración y aumenta el número de funcionalidades.

1.10 Otras tecnologías

- **Framework de desarrollo Qt**

Qt es marco de trabajo multiplataforma para el desarrollo de aplicaciones e interfaces de usuario. Es producido por la división de software Qt de Nokia. Utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación. Está distribuido bajo los términos de la licencia LGPL y la comercial. Incluye herramientas de desarrollo integradas y librerías y un IDE multiplataforma. Garantiza la portabilidad entre sistemas operativos embebidos y de escritorio. Brinda la posibilidad de trabajar con hilos independientemente del sistema operativo, posee soporte para aplicaciones orientadas a componentes.

- **LibVLC**

LibVLC es la interfaz de programación externa del VLC media player, que permite a los desarrolladores crear aplicaciones mediante las características de VLC. Es una organización sin fines de lucro, integrado por desarrollos voluntarios, multiplataforma y libre. El uso de libvlc permite realizar sistemas que sean capaces de lograr mejores prestaciones en los procesos de reproducción y transmisión de medias.

Los desarrolladores pueden utilizar la librería libvlc para aprovechar las complejas funcionalidades implementadas por VideoLAN. LibVLC se distribuye como una librería compartida, lo que permite al desarrollador de aplicaciones acceder a la funcionalidad del VideoLAN sin tener que empezar a codificarla el mismo desde cero.(Alonso Guerrero, y otros, 2009)

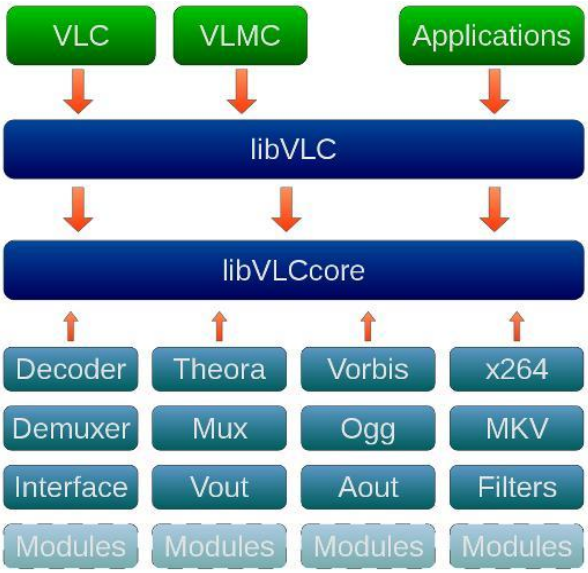

Figura 2 Esquema de composición de libVLC

1.11 Conclusiones

En este capítulo se han presentado varios conceptos, que facilitará la comprensión del trabajo, se mencionaron las aplicaciones existentes en el mundo para el monitoreo de señales digitales audiovisuales, realizándose un estudio en cuanto a las desventajas que tienen estas comparadas con la solución que se propone en este documento y fueron definidas y caracterizadas varias herramientas y tecnologías, donde se seleccionaron las más indicadas para la realización del componente.

CAPÍTULO 2: Presentación de la solución propuesta del componente para el monitoreo de las señales digitales audiovisuales

2.1 Introducción

En este capítulo se realiza una descripción de la solución propuesta, a través del modelo de dominio, y de los principales conceptos asociados a nuestro trabajo. Se identifican los requisitos funcionales y no funcionales, los casos de usos que se generan a partir de los requisitos funcionales y una explicación de cada uno de ellos a través de las descripciones textuales.

2.2 Modelo de dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en el que trabaja el sistema. (Jacobson, y otros, 2000)

No representa un conjunto de diagramas que describen clases de software u objetos de software, permite mostrar de manera visual los conceptos u objetos del mundo real más significativos de un área de interés, enlazándolos unos con otros. Se representa mediante diagramas de UML, especialmente mediante diagramas de clases.

Se aplica cuando existen flujos de información difusos, cuando no se puede establecer una estructura de los procesos de negocio, y no se pueden determinar con claridad las fronteras del mismo.

2.2.1 Diagrama del Modelo de Dominio

Se tiene en el Departamento de Señales digitales un sistema que realiza la transmisión y recepción de medias. Este sistema es utilizado por un usuario que se encarga de llevar a cabo todas las funcionalidades que brinda y velar por el correcto funcionamiento del mismo. Este usuario monitorea constantemente las medias que se están transmitiendo y modifica sus parámetros en caso de ser necesario, además puede realizar reportes con las observaciones identificadas durante su labor.

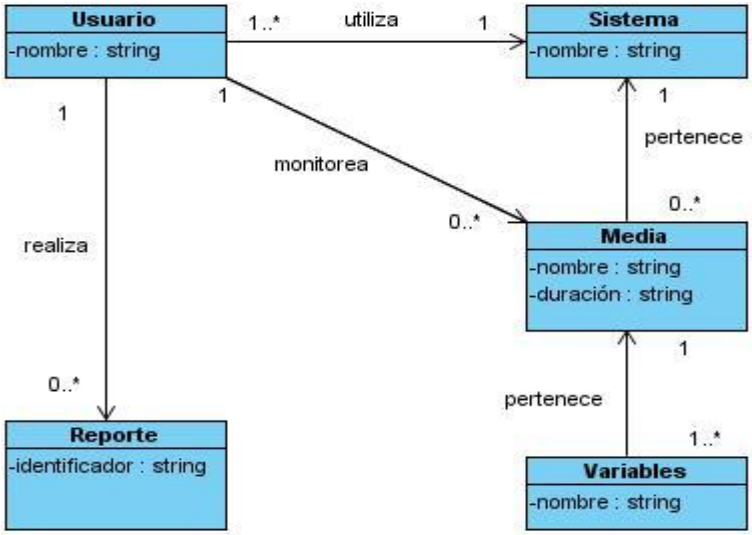

Figura 3 Diagrama Modelo de Dominio

2.2.2 Conceptos principales del entorno

Usuario: Persona que utiliza el sistema.

Reporte: Documento donde el usuario muestra los datos del video que se está transmitiendo.

Sistema: Aplicación que realiza la transmisión y recepción de las medias.

Variables: Atributos de las medias que se está transmitiendo, por ejemplo la salida de audio, los datos del canal.

Media: archivo que contiene imágenes, sonido o su combinación.

2.3 Requerimientos

En el desarrollo de un sistema lo primero que se debe realizar es la identificación de los requisitos que debe cumplir el mismo, para lograr una buena calidad en el diseño y finalmente un producto exitoso. Estos requisitos van a conformar las necesidades o deseos de un producto.

2.3.1 Requerimientos Funcionales

Los requerimientos funcionales constituyen las capacidades o condiciones que el sistema debe cumplir.

El sistema debe permitir:

RF.1 Visualizar el canal seleccionado.

RF.2 Visualizar la programación del canal seleccionado.

RF.3 Visualizar un canal a pantalla completa.

RF.4 Visualizar datos del canal.

RF.5 Visualizar salida de audio de los canales en transmisión.

RF.6 Mostrar reporte de errores.

RF.7 Insertar error.

RF.8 Modificar error.

RF.9 Eliminar error.

2.3.2 Requerimientos No Funcionales

Los requerimientos no funcionales son las cualidades o propiedades que el producto debe tener.

RNF1. Apariencia o interfaz externa.

El componente debe poseer una interfaz amigable, de fácil manejo y que posibilite el buen desempeño en la ejecución de sus funciones.

RNF2. Restricciones en el diseño e implementación.

El componente estará implementado en lenguaje C++, utilizando como IDE de desarrollo QT Creator. Para la modelación se utilizará el Visual Paradigm.

RNF3. Disponibilidad

La disponibilidad del componente debe ser continua, de 7 días por 24 horas, garantizando ante una posible falla en la transmisión dar una respuesta inmediata.

RNF4. Usabilidad

El componente podrá ser usado por usuarios que posean conocimientos básicos sobre computadoras, por lo que todas sus funcionalidades deben ser claras y sencillas.

RNF5. Software.

Se debe tener instaladas las librerías del framework Qt y VLC, el componente debe funcionar sobre Sistema Operativo Linux, a partir de la distribución Ubuntu 10.10.

RNF6. Hardware

Para la implantación del componente se necesitan computadoras con las siguientes prestaciones:

PC Cliente:

- 1gb de RAM.
- 1 GB de espacio libre en disco.
- Microprocesador Dual Core a 3.0 GHz.

Servidor de Base de Datos:

- RAM = 256 MB CPU = 1.0 GHz

2.4 Descripción del sistema propuesto. Modelos de Casos de Uso del Sistema

A continuación se describe el modelo de casos de uso del componente, haciendo uso del lenguaje de modelado UML, se representan mediante un diagrama las funcionalidades del componente, y por su importancia se definirán los actores y los casos de uso que representarán las responsabilidades del mismo.

2.4.1 Descripción de los actores

Un actor es un rol que cumple un usuario, puede intercambiar información o puede ser un recipiente pasivo de información y representa a un ser humano, a un software o a una máquina que interactúa con el sistema. (Jacobson, y otros, 2000)

Actor	Descripción
Usuario	Rol responsable de velar por el funcionamiento apropiado del componente, configurarlo y utilizar todas las funcionalidades que permite el componente.

Tabla 1 Descripción de los actores

2.4.2 Casos de Uso del Sistema

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la funcionalidad total del sistema.(Jacobson, y otros, 2000)

CU-1	Visualizar media.

Actor	Usuario
Descripción	Visualiza la señal y la salida de audio del canal deseado.
Referencia	RF.1
CU-2	**Visualizar programación.**
Actor	Usuario
Descripción	Se visualiza la programación correspondiente al canal seleccionado.
Referencia	RF.2
CU-3	**Visualizar pantalla completa.**
Actor	Usuario
Descripción	Se visualiza la media deseada a pantalla completa.
Referencia	RF.3
CU-4	**Visualizar datos.**
Actor	Usuario
Descripción	Se visualizan los datos del canal deseado.
Referencia	RF.4
CU-5	**Mostrar reporte.**
Actor	Usuario
Descripción	Se muestra un reporte con los errores ocurridos en la transmisión.
Referencia	RF.5
CU-6	**Gestionar error.**
Actor	Usuario
Descripción	Se insertan, modifican y eliminan los errores ocurridos en la transmisión.
Referencia	RF.6

Tabla 2 Casos de uso del sistema

2.4.3 Diagramas de Casos de Uso del Sistema

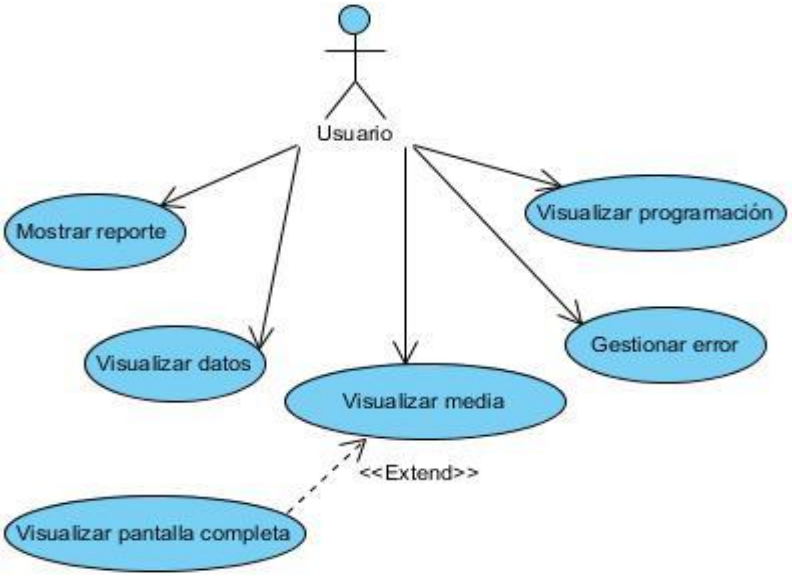

Figura 4 Diagrama de casos de uso del sistema

2.4.4 Descripción de los Casos de Uso (CU)

2.4.4.1 Descripción del CU Visualizar media

Objetivo	Visualizar media.	
Actores	Usuario	
Resumen	Caso de uso que visualiza la señal y la salida de audio del canal deseado.	
Prioridad	Crítico	
Precondiciones	Se debe estar transmitiendo el canal.	
Postcondiciones	La media fue visualizada.	
Flujo de eventos		
Flujo básico: Visualizar canal.		
Actor		**Sistema**
1. El usuario selecciona la opción visualizar media.		2. El sistema muestra los canales a seleccionar.
3. El usuario selecciona el canal deseado y oprime el botón aceptar.		4. El sistema visualiza la señal y la salida de audio del canal

	seleccionado.

Tabla 3 Descripción del CU Visualizar media

2.4.4.2 Descripción del CU Gestionar error

Objetivo	Gestionar error.
Actores	Usuario
Resumen	Caso de uso que permite insertar, eliminar y modificar los errores detectados en la transmisión.
Prioridad	Crítico
Precondiciones	Se debe estar transmitiendo algún canal.
Postcondiciones	El error fue gestionado.

Flujo de eventos	
Flujo básico: Gestionar error.	
Actor	**Sistema**
1. El usuario selecciona la opción de gestionar error.	Permite realizar varias acciones: - Insertar error. - Modificar error. - Eliminar error.
2.	El sistema actualiza los datos de los errores.

Sección 1: "Insertar"		
Flujo básico: Gestionar error		
	Actor	**Sistema**
1.	Selecciona la opción insertar.	Muestra el formulario para entrar los datos del error a insertar.
2.	Entra los datos y oprime el botón aceptar.	Adiciona el error.

Sección 2: "Eliminar"		
Flujo básico: Gestionar error.		
	Actor	**Sistema**
1.	Selecciona el error a eliminar y oprime el botón aceptar.	Elimina el error.
Sección 3: "Modificar"		
Flujo básico: Gestionar error.		
	Actor	**Sistema**
1.	Entra los datos del error a modificar y oprime el botón aceptar.	Modifica el error.

Tabla 4 Descripción del CU Gestionar error

Ver las demás descripciones de los demás casos de uso en el Anexo 1.

2.5 Conclusiones

En este capítulo se realizó una descripción de la solución propuesta al problema inicialmente planteado. Se determinó el modelo de domino para una mejor comprensión del entorno donde se desarrollará el componente, se identificaron los requisitos funcionales y no funcionales quedando de esta manera señaladas las funcionalidades que el sistema de cumplir y las cualidades que debe tener, se seleccionaron los actores y casos de usos del componente especificando sus interacciones mediante un Diagrama de Casos de Uso. Además, se realizó una descripción detallada de los casos de usos para obtener mayor conocimiento de cómo funcionará en general toda la aplicación. Teniendo en cuenta todas estas características se puede comenzar a construir el sistema poniendo en práctica el cumplimiento de los requisitos tanto funcionales como no funcionales planteados en este capítulo.

CAPÍTULO 3: Diseño de la solución propuesta del componente para el monitoreo de las señales digitales audiovisuales

3.1 Introducción

En este capítulo se realiza el diseño del componente a partir de la descripción de las funcionalidades abordadas en el capítulo anterior. Se representan los diagramas de clases y diagramas de interacción para una mejor comprensión del sistema que se propone, se describe la arquitectura del sistema y los patrones utilizados y se presenta el diseño de la base de datos mediante los diagramas Entidad-Relación y el Diagrama de Clases Persistentes.

3.2 Patrones de Diseño

Los patrones constituyen modelos a seguir ya que son soluciones a un problema en un contexto determinado, capturan las experiencias existentes y probadas para promover buenas prácticas.

Establecen la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Para el diseño de la aplicación se utilizaron varios patrones de diseño.

3.2.1 Patrones GRASP

Para la asignación de responsabilidades se tendrán en cuenta algunos de los Patrones GRASP. Dentro de ellos el patrón Experto, el cual establece que se debe asignar una responsabilidad al experto en información que sería a la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Además, se utilizará el patrón Creador, que establece la necesidad de asignarle a una clase la responsabilidad de crear una instancia de otra clase cuando agregue los objetos de la clase, los contenga, registre las instancias de estos objetos o los utilice.

Para fomentar la reutilización de código se tendrá en cuenta el patrón Bajo Acoplamiento que establece una menor dependencia entre clases y para mejorar la claridad y facilidad del diseño, simplificar el mantenimiento y las mejoras en funcionalidad se utilizará el patrón Alta Cohesión que expresa que se debe asignar una responsabilidad de modo que la cohesión siga siendo alta, la cohesión no es más que la medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.

También se tendrá en cuenta para manejar y controlar los eventos del sistema el patrón Controlador. De este modo se logra separar la lógica de negocio de la capa de presentación, un mayor control sobre el sistema y se favorece la reutilización de código.

3.2.2 Patrones de Arquitectura

3.2.2.1 Arquitectura en 3 Capas

La arquitectura en 3 capas consiste en separar un proyecto en Capa de Presentación, Capa de Negocio y Capa de Datos.

Capa de presentación o interfaz de usuario: Esta capa está formada por los formularios y los controles que se encuentran en los formularios. Capa que interactúa con el usuario, es decir, son las vistas y las clases controladoras relacionadas con cada módulo.

Capa de negocio: Esta capa está formada por las operaciones lógicas que se realizaran sobre las entidades del negocio, objetos que van a ser manejados o consumidos por toda la aplicación.

Capa de acceso a datos: Esta capa contiene las clases que interactúan directamente con la base de datos, las operaciones que estas realizan son transparente para la capa de negocio, en este caso son la clase modelo principal, la clase que permite la conexión con la base de datos.

Esto permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

Ventajas de esta Arquitectura:

- Desarrollos paralelos (en cada capa).
- Aplicaciones más robustas debido al encapsulamiento.
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica).
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad).
- En caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.
- Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buena escalabilidad, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

3.2.2.2 Modelo Cliente/Servidor

Las características básicas de una arquitectura Cliente/Servidor son:

• Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, módems, etc.

• Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.

• Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores.

• La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.

• Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.

• El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.(Márquez Avendaño, y otros, 2004)

3.2.2.3 Modelo vista controlador (MVC)

Es un patrón de diseño de arquitectura de software usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de conceptos para que el desarrollo esté estructurado de una mejor manera, facilitando la programación en diferentes capas de manera paralela e independiente. (Rivera López, 2008)

Sugiere la separación del software en 3 capas:

Modelo: es el objeto que representa los datos del programa, maneja el comportamiento de los mismos, responde a requerimientos de información sobre su estado y responde a instrucciones de cambiar el estado.

Vista: Maneja la visualización de la información.

Controlador: Controla el flujo entre la vista y el modelo.

Principales Ventajas:

- Permite tener un completo control sobre el comportamiento de una aplicación.
- Soporte de vistas múltiples, es decir, dado que la vista se encuentra aislada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente.
- Las interfaces tienden a cambiar más rápido que las reglas de negocios. Agregar nuevos tipos de vistas no afectan al modelo.
- Permite un diseño modular, posibilitando que los diseñadores y los desarrolladores trabajen conjuntamente.
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Este patrón se centra en la separación del modelo y la vista, mientras que el controlador es el encargado de relacionarlos. Su principal característica es aislar la vista del modelo. El siguiente diagrama representa su ciclo de vida.

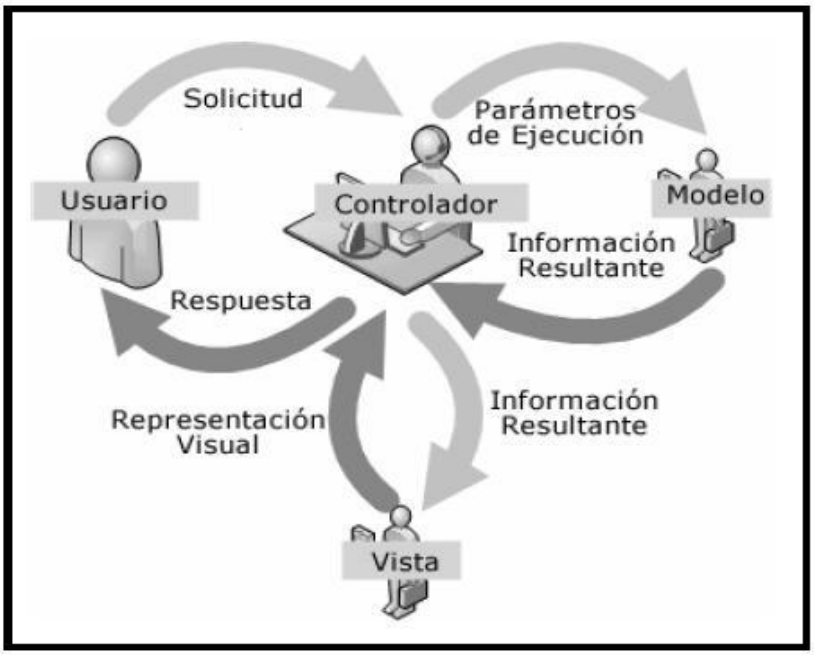

Figura 5 Ciclo de vida de la arquitectura Modelo Vista Controlador

3.2.2.4 ¿Por qué arquitectura Modelo Vista Controlador?

Para el desarrollo de la arquitectura se tendrá en cuenta el patrón Modelo Vista Controlador ya que esta separa la lógica de negocio (el modelo) y la presentación (la vista) lo que consigue un mantenimiento más sencillo de las aplicaciones. Esta separación permite construir y probar el modelo independientemente de la representación visual, es más sencillo agregar múltiples representaciones de los datos o información, dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente, además teniendo una clase encargada de controlar y dirigir los eventos del sistema se tendrá una organización de código separado por sus funciones.

3.3 Diagramas de clase del diseño

En el diseño modelamos la aplicación y encontramos su forma para que soporte todos los requisitos, incluyendo los no funcionales.

Un diagrama de clase muestra un conjunto de clases, interfaces, colaboraciones y sus relaciones. Este diagrama es el que más se encuentra en los sistemas de modelado orientado a objetos. Los diagramas de clases se utilizan para modelar la visión estática de un sistema. Esta visión soporta los requisitos funcionales del sistema, en concreto, los servicios que el sistema debería proporcionar a sus usuarios finales.(Otero Vidal, 2003)

3.3.1 Diagrama General de Clases del Diseño

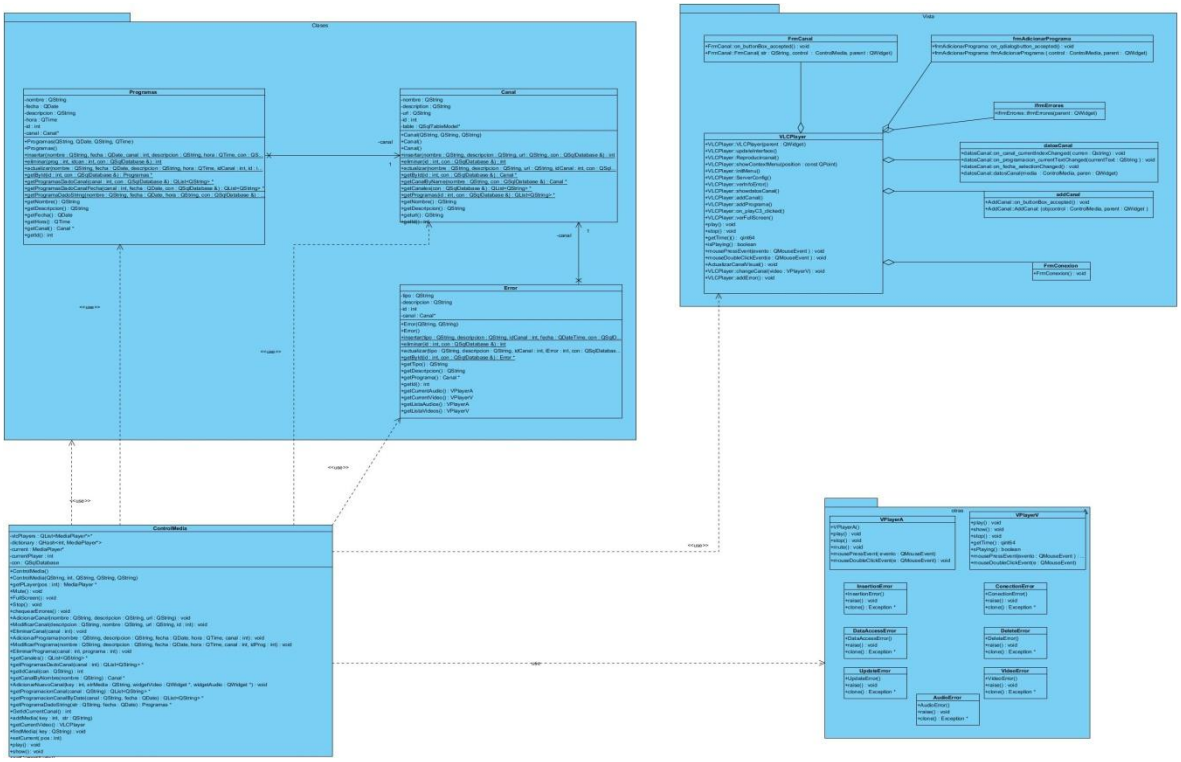

Figura 6 Diagrama General de Clases del Diseño

3.3.2 Diagrama de Clases del Diseño del CU Visualizar media

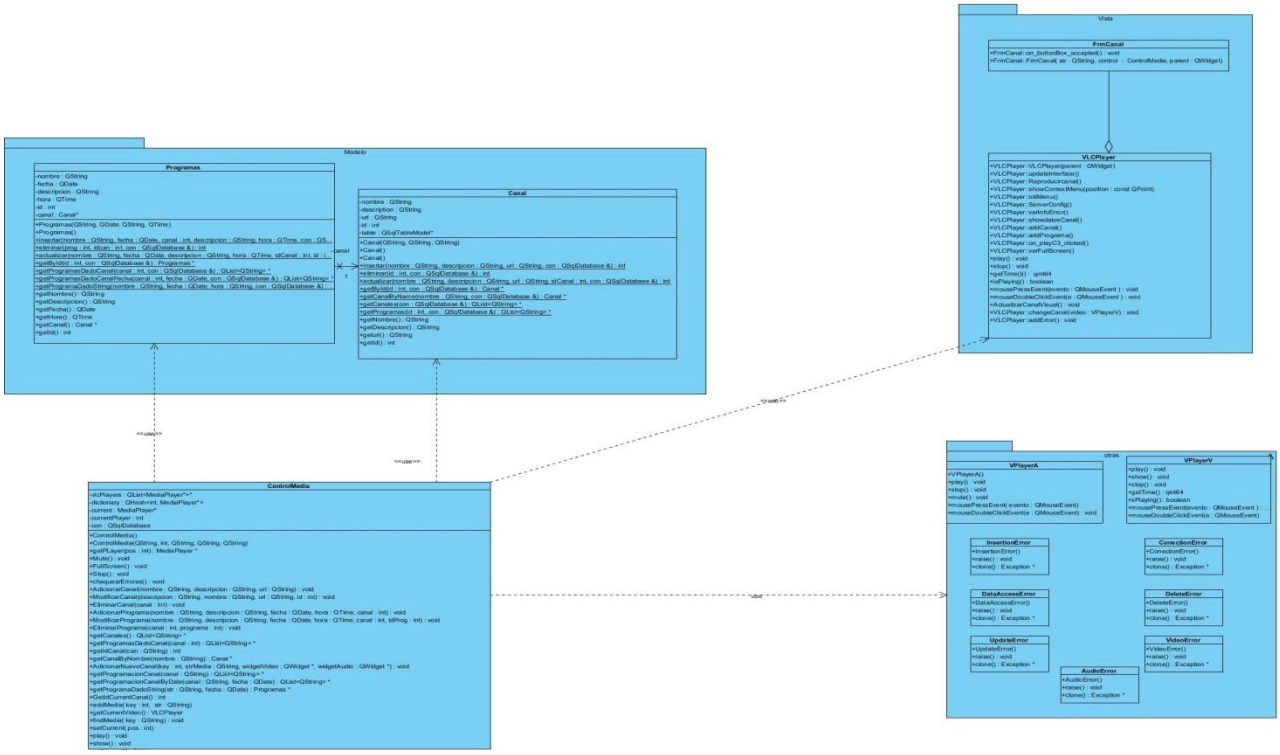

Figura 7 Diagrama de Clases del Diseño del CU Visualizar media

3.3.3 Diagrama de Clases del Diseño del CU Gestionar error

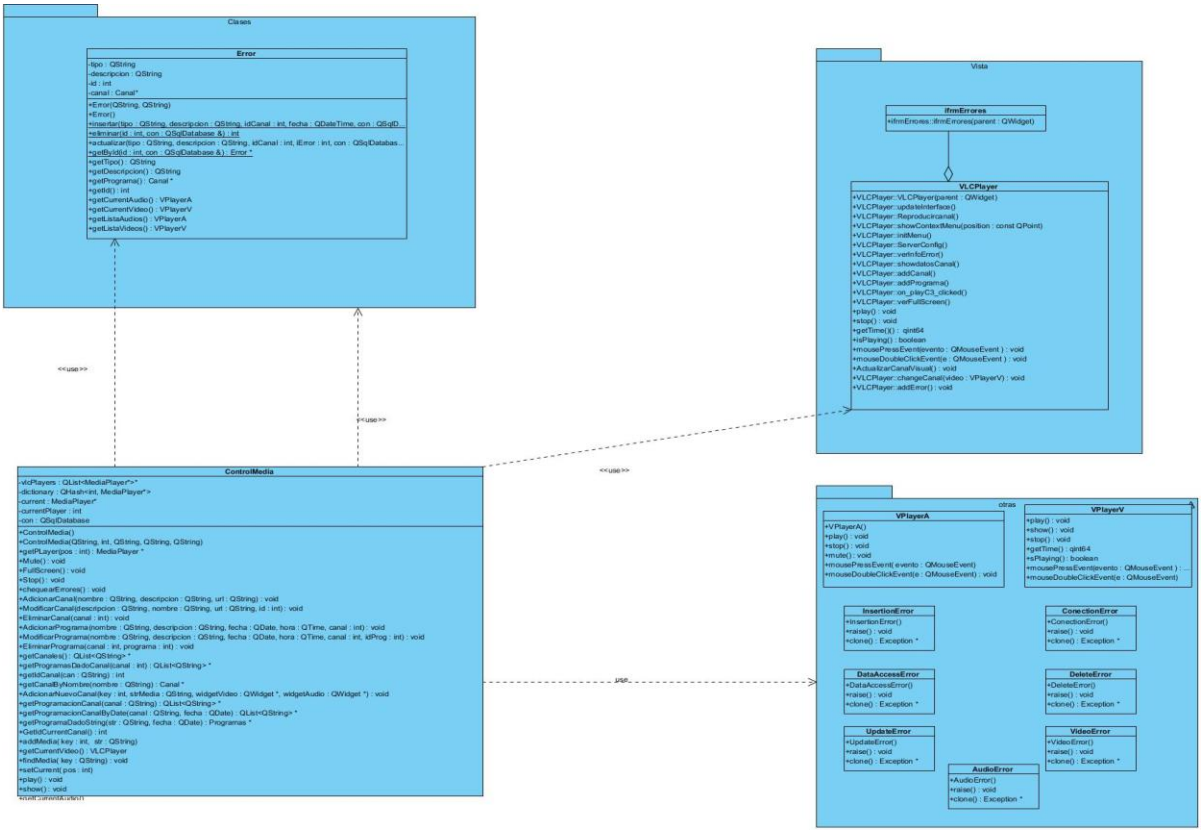

Figura 8 Diagrama de Clases del Diseño del CU Gestionar error

Ver los demás diagramas de clases del diseño en el Anexo 2.

3.4 Diagramas de Interacción

La ejecución de un software orientado objeto consiste en un encadenamiento de operaciones y cambios de estado de objeto, el cual, a su vez, consiste en que durante la ejecución de una operación o durante una transición se llaman operaciones sobre otros objetos(o sobre el mismo) y se envían señales que provocan otras transiciones. Así, se puede describir el funcionamiento de los casos de uso y de operaciones complejas. En el UML esta acción se lleva cabo mediante los denominados diagramas de interacción. Una interacción es la especificación del comportamiento de un caso de uso u operaciones en

términos de secuencia de intercambio de mensajes entre objetos. Estos mensajes contienen estímulos, que pueden ser peticiones de ejecución de operaciones o señales.(Falgueras, 2003)

3.4.1 Diagramas de Secuencia

Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes.

3.4.1.1 Diagrama de Secuencia. CU Visualizar media

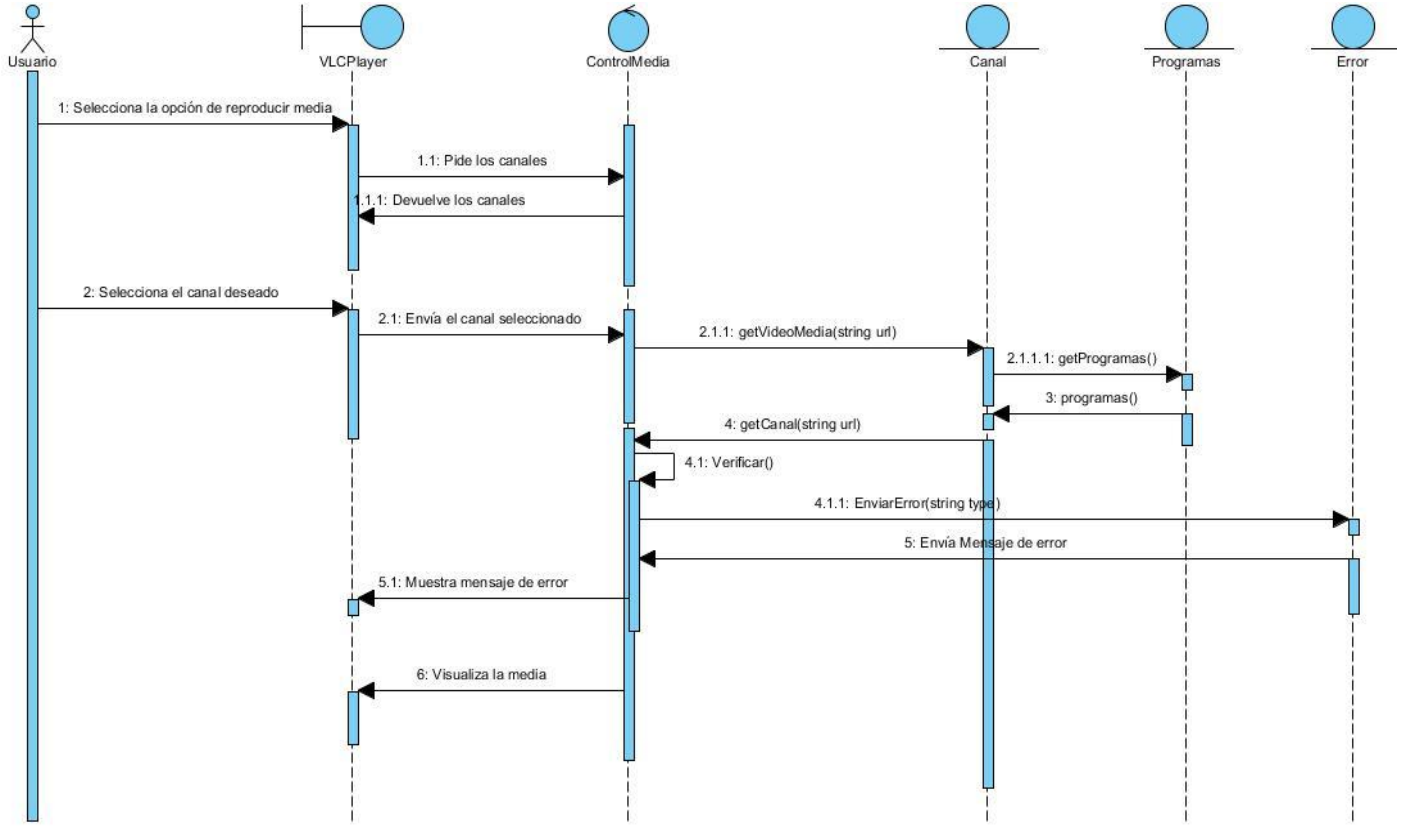

Figura 9 Diagrama de Secuencia CU Visualizar Media

3.4.1.2 Diagrama de Secuencia. CU Gestionar error

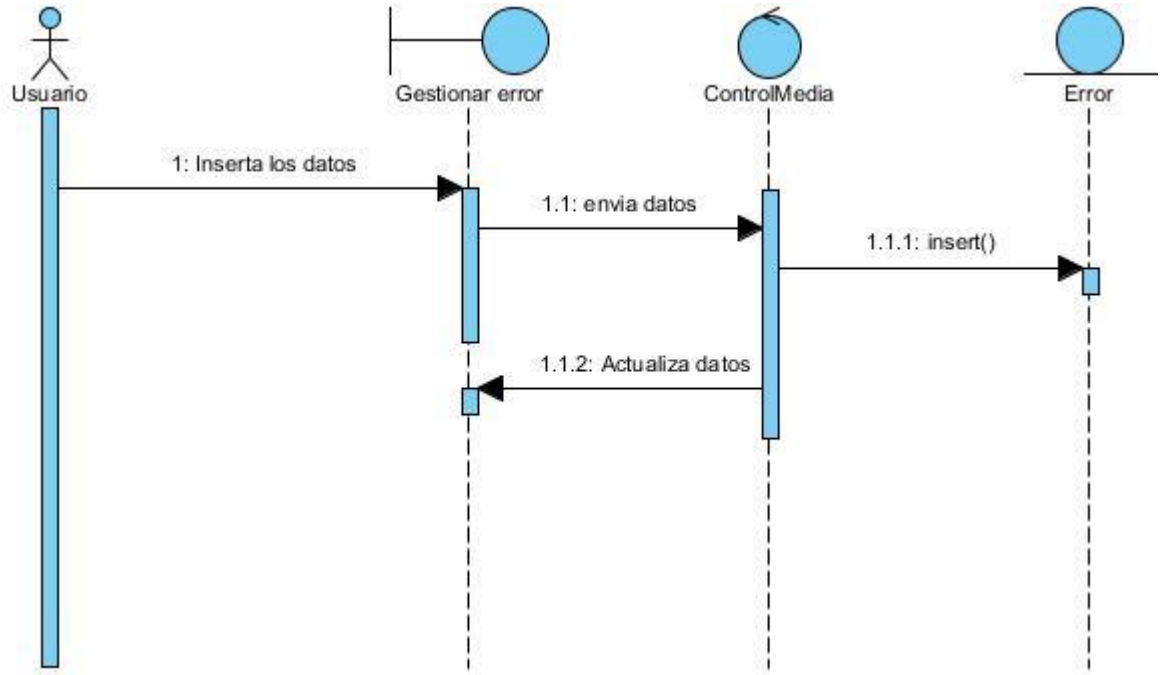

Figura 10 Diagrama de Secuencia CU Insertar error.

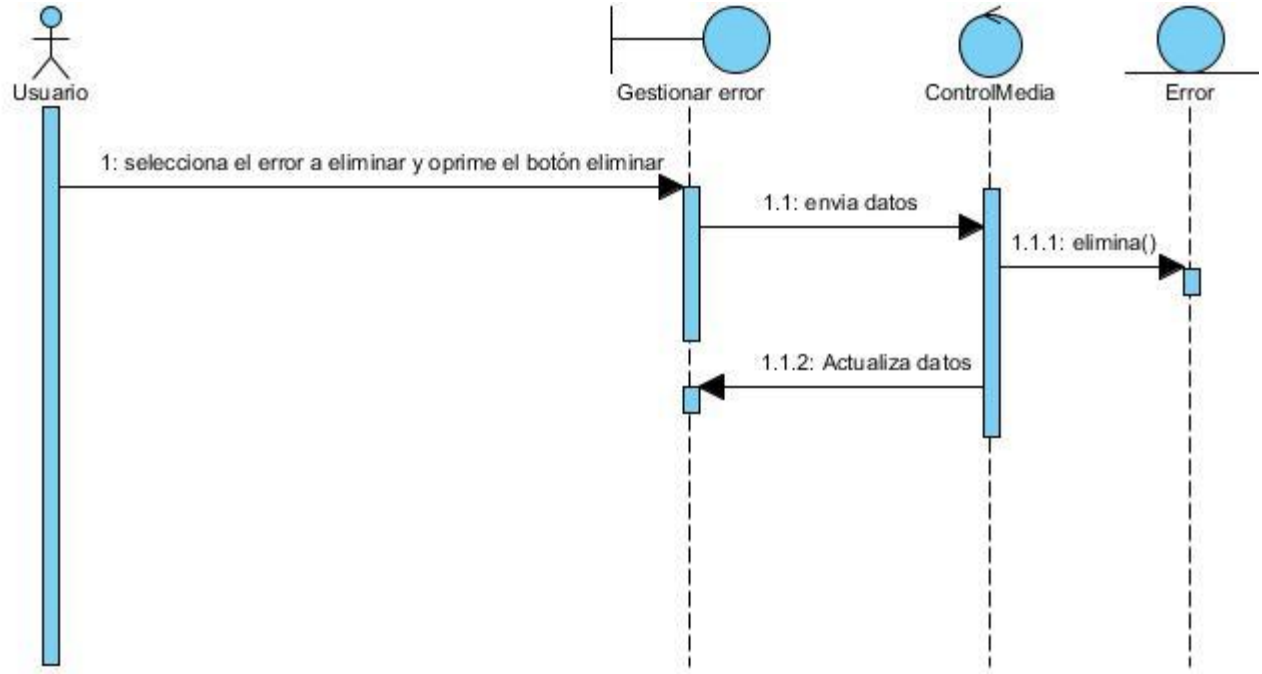

Figura 11 Diagrama de Secuencia CU Eliminar error.

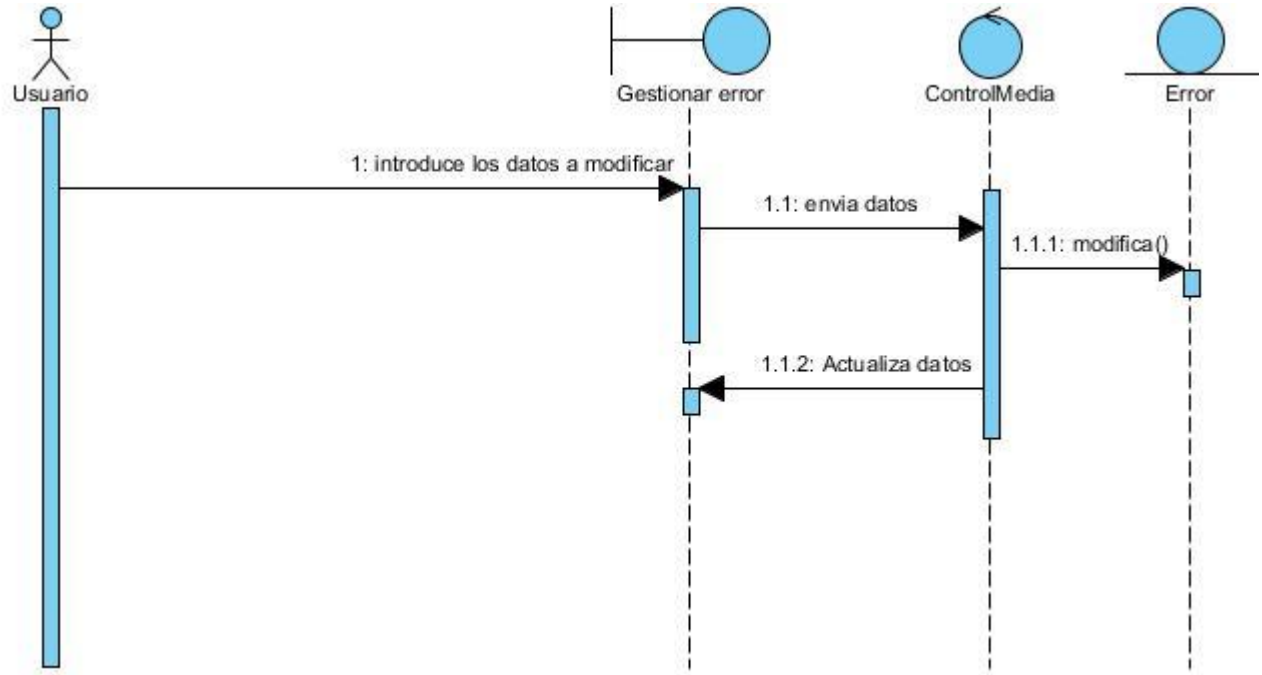

Figura 12 Diagrama de Secuencia CU Modificar error.

Ver los demás diagramas de secuencia en el Anexo 3.

3.4.2 Diagramas de Colaboración

Un diagrama de colaboración es un diagrama de interacción que destaca la organización estructural de los objetos que envían y reciben mensajes.

3.4.2.1 Diagrama de Colaboración. CU Visualizar media

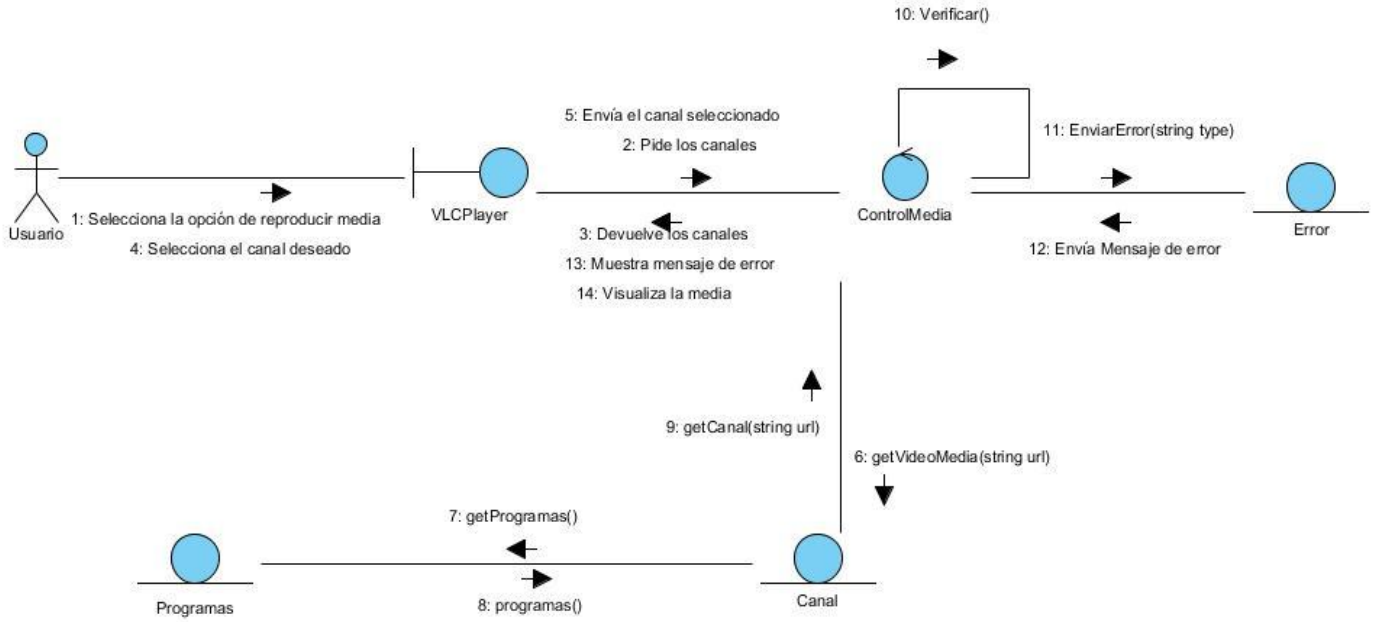

Figura 13 Diagrama de Colaboración CU Visualizar media.

3.4.2.2 Diagrama de Colaboración. CU Gestionar error

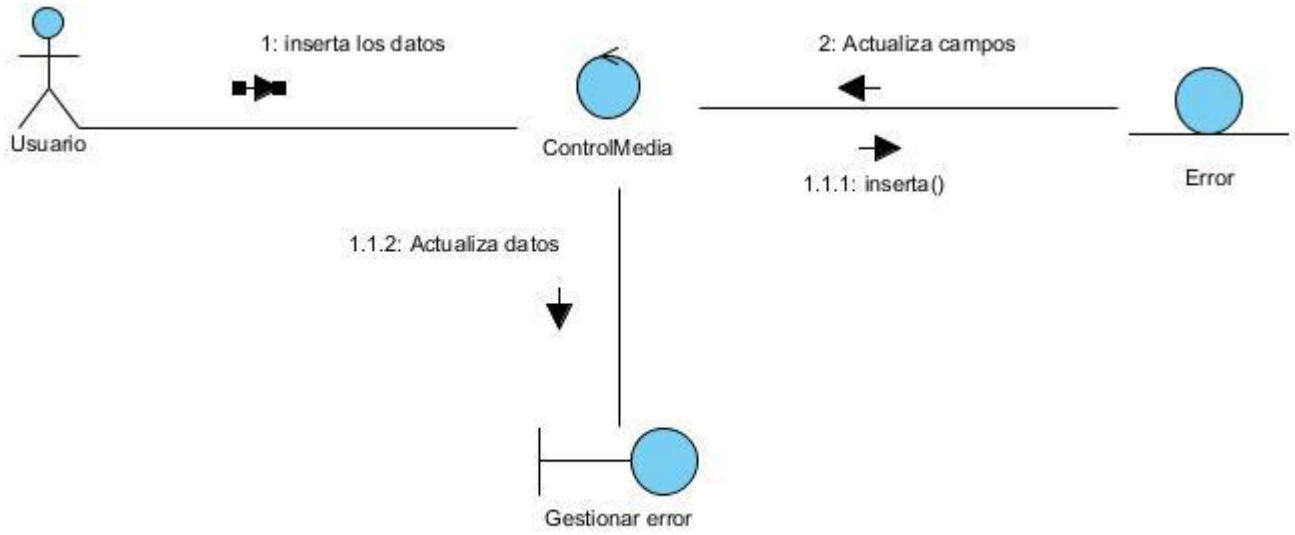

Figura 14 Diagrama de Colaboración CU Insertar error.

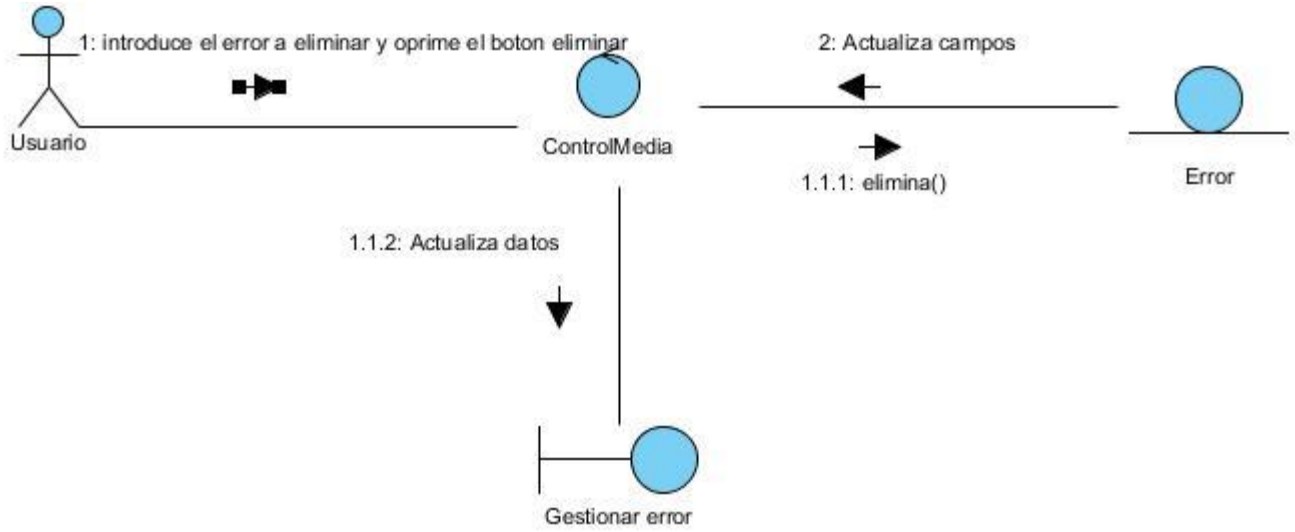

Figura 15 Diagrama de Colaboración CU Eliminar error.

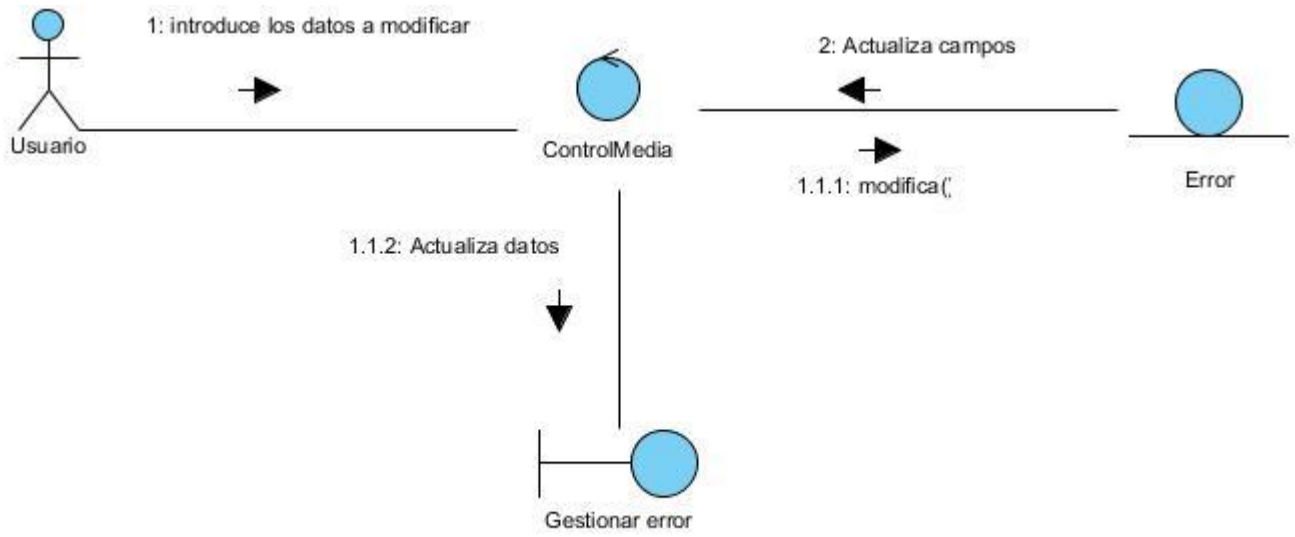

Figura 16 Diagrama de Colaboración CU Modificar error.

Ver los demás diagramas de colaboración en el Anexo 3.

3.5 Diseño de la Base de Datos

Para lograr un correcto funcionamiento del componente también es necesario un buen diseño y funcionamiento de la base de datos a utilizar. A continuación se muestra el diagrama de entidad relación y el diagrama de clases persistentes correspondiente al diseño de la BD.

El modelo entidad relación representa a la realidad a través de un esquema gráfico empleando la terminología de entidades, que son objetos que existen y son los elementos principales que se identifican en el problema a resolver con el diagramado y se distinguen de otros por sus características particulares denominadas atributos, el enlace que rige la unión de las entidades está representada por la relación del modelo.(López Meléndez, 2009)

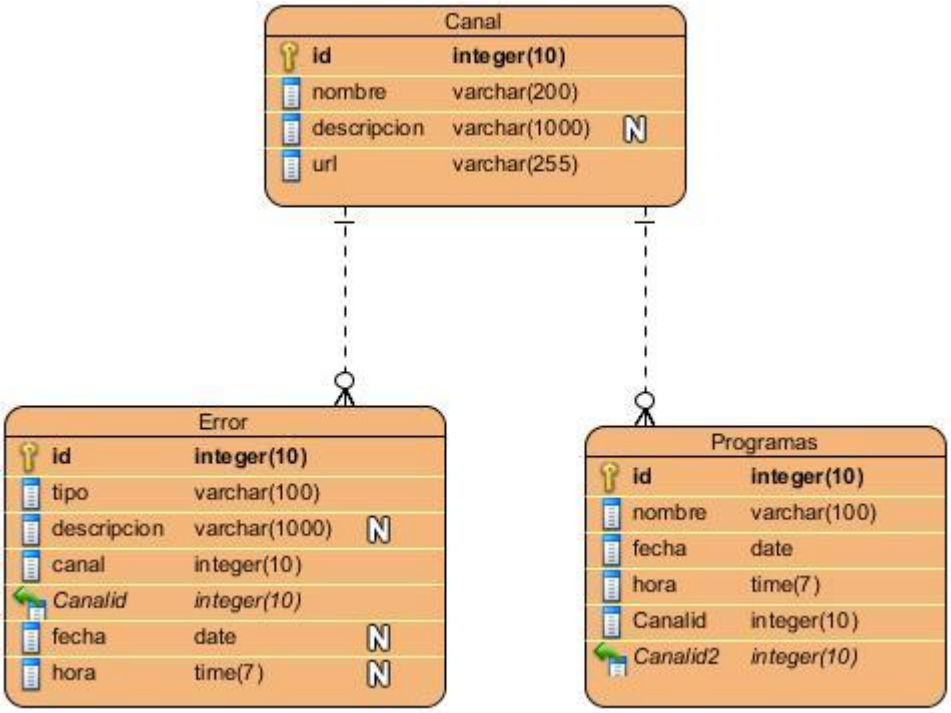

Figura 17 Diagrama entidad relación

El diagrama de clases persistentes muestra las relaciones existentes entre aquellas clases que implementan las entidades del problema de negocio, manteniendo su valor en un espacio y tiempo determinado.

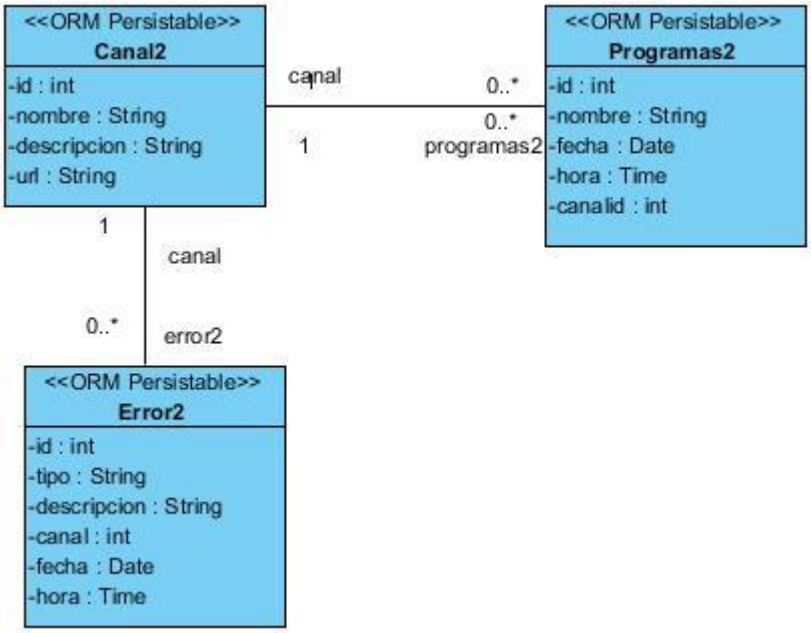

Figura 18 Diagrama de clases persistentes

3.6 Conclusiones

En el presente capítulo se ha descrito el diseño de la solución propuesta a través de los distintos artefactos que define RUP para la construcción de aplicaciones. Se describieron los patrones de diseño a seguir para lograr una mayor claridad y sencillez en la implementación del componente, se presentaron los diferentes diagramas de clases del diseño permitiendo visualizar de manera clara, los procesos a automatizar y se realizó además el diseño de la base de datos para un mayor entendimiento del funcionamiento de la aplicación representado a través del diagrama entidad relación y el diagrama de clases persistentes.

CAPÍTULO 4: Implementación y proceso de pruebas del componente para el monitoreo de las señales digitales audiovisuales

4.1 Introducción.

En este capítulo se describe cómo los elementos del modelo del diseño se implementan en términos de componentes. Además, se realiza el proceso de pruebas al componente, se valida la solución propuesta a través de la prueba seleccionada para garantizar el correcto funcionamiento de la aplicación y se realiza el diseño de los casos de prueba, obteniendo un resultado de estas pruebas realizadas al componente.

4.2 Modelo de Implementación

En la implementación se describe cómo los elementos del Modelo de Diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el Modelo de Despliegue.

4.2.1 Diagrama de despliegue

El diagrama de despliegue muestra las relaciones entre el hardware y el software en el sistema final. Se representa como un grafo de nodos unidos por conexiones de comunicación.

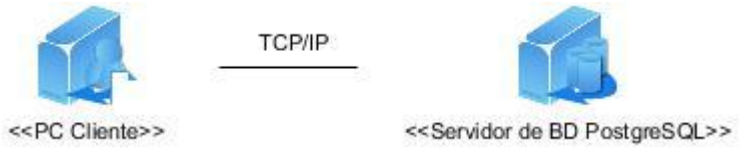

Figura 19 Diagrama de despliegue

A continuación se describen los componentes y protocolos presentes en el diagrama de despliegue.

Figura 20 Nodo PC para acceder al componente

Este nodo representa una PC cliente desde la cual se puede acceder al componente para monitorear las señales digitales.

Figura 21 Nodo servidor de BD

En este nodo se estará ejecutando el servidor PostgreSQL con la base de datos del sistema.

Protocolos:

ADO: Tecnología que permite conectarse al servidor de BD.

4.2.2 Diagrama de componentes

Los Diagramas de componentes son diagramas que muestran un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones. Son usados para modelar el código fuente, la base de datos física, la versión ejecutable y bibliotecas.

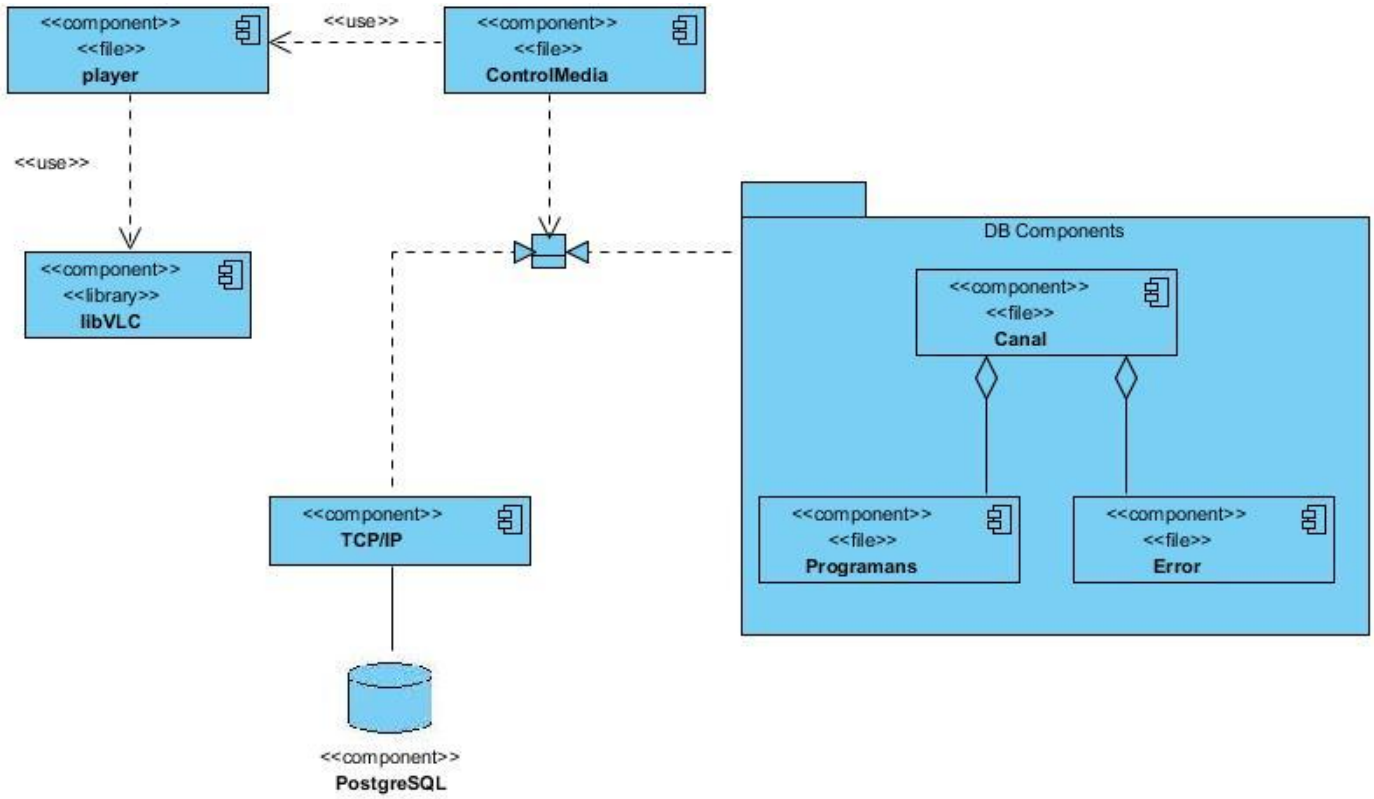

Figura 22 Diagrama de Componentes.

4.3 Prueba de software

Concretamente la prueba de software se puede definir como una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas, registrándose los resultados obtenidos. El objetivo de las pruebas no es asegurar la ausencia de defectos en un software, únicamente puede demostrar que existen defectos en el software. El objetivo es pues, diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. (Moreno, 2005)

4.4 Objetivos de las pruebas

Glem Myers establece varias normas que pueden servir adecuadamente como objetivos de la prueba:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.

- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Los objetivos anteriores, suponen un cambio importante desde un punto de vista, ya que la idea normal es que una prueba tiene éxito si no descubre errores. El objetivo de la prueba es: "diseñar pruebas que saquen a la luz diferentes clases de errores con la menor cantidad de tiempo y espacio". (Pressman, 1998)

La prueba de software constituye un proceso para validar y verificar que los requisitos especificados por los clientes fueron cumplidos centrándose en la búsqueda de errores.

4.5 Técnicas de prueba de software

Las técnicas de evaluación dinámica o prueba proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas. Estas técnicas se agrupan en:

- Técnicas de caja blanca o estructural, que se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- Técnicas de caja negra o funcionales, que realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. (Moreno, 2005)

4.5.1 Prueba de Caja Negra

La técnica seleccionada para aplicar al componente es la prueba de Caja Negra, la misma se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene.

Se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Muchos autores consideran que estas pruebas permiten encontrar: (Pressman, 1998) (Beiser, 1995).

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están: (Pressman, 2000).

- Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

4.5.1.1 Partición equivalente

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. (Software, 2011)

La técnica escogida para aplicar al componente es la Técnica de la Partición de Equivalencia. Dentro del método de Caja Negra ésta técnica es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

4.6 Diseños de Casos de Prueba

4.6.1 Diseño de Caso de Prueba del CU Visualizar media

- **Descripción General**

Caso de uso que visualiza la señal y la salida de audio del canal deseado.

- **Condiciones de Ejecución**

Se debe estar transmitiendo el canal.

- **Secciones a probar en el Caso de Uso**

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: "Visualizar media."	EC 1.1: Visualizar media.	El usuario selecciona la opción visualizar media. El sistema muestra los canales a seleccionar. El usuario selecciona el canal deseado y oprime el botón aceptar. El sistema visualiza la señal y la salida de audio del canal seleccionado.	Interfaz principal/Botón Canales/Botón Reproducir

Tabla 5 Secciones a probar en el CU Visualizar media

- **Matriz de Datos**

SC 1 Visualizar media

ID del escenario	Escenario	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Visualizar media.	El sistema visualiza en pantalla la media deseada.	Satisfactorio.

Tabla 6 Matriz de Datos. SC 1 Visualizar media

Diseño de Caso de Prueba del CU Gestionar error.

- **Descripción General.**

Caso de uso que permite insertar, eliminar y modificar los errores detectados en la transmisión.

- **Condiciones de Ejecución.**

Se debe estar transmitiendo algún canal.

- **Secciones a probar en el Caso de Uso.**

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: "Insertar error."	EC 1.1: Insertar error.	El usuario selecciona la opción Insertar error.(Botón Gestionar error) El sistema muestra la interfaz para introducir los datos. El usuario introduce los datos y oprime el botón aceptar. El sistema inserta el error.	Interfaz principal/Botón Gestionar error.
SC 2: "Eliminar error."	EC 2.1: Eliminar error.	El usuario selecciona la opción Eliminar error. (Botón Gestionar error) El sistema muestra los errores a eliminar. El usuario selecciona el error que desea eliminar y oprime el botón eliminar. El sistema elimina el error.	Interfaz principal/Botón Gestionar error.

SC 3: "Modificar error."	EC 3.1: Modificar error.	El usuario selecciona la opción Modificar error. (Botón Gestionar error) El sistema muestra los errores a modificar. El usuario selecciona el error que desea modificar, modifica sus datos y oprime el botón aceptar. El sistema actualiza el error.	Interfaz principal/Botón Gestionar error.

Tabla 7 Secciones a probar en el CU Gestionar error

- **Matriz de Datos**

SC 1 Gestionar error.ID del escenario	**Escenario**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1	Insertar error.	El sistema inserta el error satisfactoriamente.	Satisfactorio.
EC 2.1	Eliminar error.	El sistema elimina el error satisfactoriamente.	Satisfactorio.
EC 3.1	Modificar error.	El sistema modifica el error satisfactoriamente.	Satisfactorio.

Tabla 8 Matriz de Datos. SC 1 Gestionar error

Ver los demás diseños de Casos de prueba en el Anexo 4.

4.7 Conclusiones

En este capítulo se presentaron los diagramas de despliegue y componentes para lograr un mejor entendimiento de la distribución física y lógica del sistema. Se realizó la validación de los resultados obtenidos en la investigación, con la misma se pudo apreciar que cada uno de los requisitos planteados por los clientes fueron desarrollados de forma satisfactoria por lo que se cumple con los objetivos trazados al inicio del trabajo y se garantiza la correcta funcionalidad del componente.

CONCLUSIONES GENERALES

Después de haber analizado los resultados obtenidos con la elaboración del presente trabajo y la implementación del componente para el monitoreo de señales digitales audiovisuales se llegaron a las siguientes conclusiones:

- Se llevaron a cabo los diferentes flujos de trabajo que propone la metodología utilizada, dándole cumplimiento a las tareas de la investigación propuestas.
- Se realizó una búsqueda de las tecnologías y herramientas más utilizadas en la solución de aplicaciones relacionadas con el objeto de estudio, lo que permitió seleccionar las adecuadas para el desarrollo del componente permitiendo que el producto final tenga la calidad requerida.
- Se realizó un análisis sobre las principales aplicaciones para el monitoreo de las señales digitales audiovisuales existentes en la actualidad con el objetivo de encontrar una solución para el problema planteado pero ninguna satisfacía las necesidades porque la mayoría tienen en su funcionamiento gran dependencia del fabricante por lo que a partir de ahí se defendió la idea propuesta.
- El sistema implementado aumenta la cantidad de componentes disponibles en el Departamento de Señales Digitales, lo cual mejora la capacidad de respuesta del Departamento de Señales Digitales ante nuevos proyectos que necesiten del componente.
- Al término de la investigación se logró la implementación de la versión 1.0 del componente para el monitoreo de señales digitales audiovisuales. El mismo cumple todos los requisitos planteados al inicio del trabajo para su desarrollo por lo que se convierte en una herramienta cuyas funcionalidades básicas muestran los resultados esperados.

Por todo lo anterior se concluye que los objetivos propuestos para el presente trabajo han sido cumplidos satisfactoriamente. El componente está listo para su utilización y con la capacidad de mejorar ampliamente en funcionalidades adaptándolo a las condiciones del cliente que lo solicite.

RECOMENDACIONES

Con el objetivo de ampliar las funcionalidades del componente desarrollado se recomienda:

- Permitir de forma dinámica la gestión de errores.

- Continuar perfeccionando el componente desarrollado a partir de los nuevos requisitos que puedan surgir como resultado de su explotación.

REFERENCIAS BIBLIOGRÁFICAS

Alonso Guerrero, Zorilin y Leyva González, Genry. 2009. ***Sistema de Captura y Transcripción de Audio.*** **La Habana : s.n., 2009.**

Beck, Kent. 1999. ***Beck, K. Extreme Programming Explained.Embrace Change.*** **1999.**

Calderón, Amaro, y otros. 2007. ***Metodologías Ágiles.*** **Trujillo Perú : s.n., 2007.**

Castañeda, Gerardo Gerardo. 2006. ***LOS ARCHIVOS AUDIOVISUALES EN LAS REDES DIGITALES DE COMUNICACIÓN PARA LA EDUCACIÓN Y LA CULTURA.*** **Madrid, España. : s.n., 2006. ISBN (PUBLICACIÓN EN CD-ROM): 84-369-4135-7.**

Cerezo López, Yolanda, Peñalba Rodríguez, Olga y Caballero Roldán, Rafael. 2007. ***Iniciación a la programación en C#: un enfoque práctico.*** **2007.**

Cruz Chávez, Marco Antonio. 2010. **Universidad Autónoma del Estado de Morelos. [En línea] 4 de 3 de 2010. [Citado el: 17 de 1 de 2011.] http://www.uaem.mx/posgrado/mcruz/cursos/miic/MySQL.pdf.**

Entorno Virtual de Aprendizaje. 2010. **Entorno Virtual de Aprendizaje. [En línea] 2010. [Citado el: 24 de 10 de 2010.] http://eva.uci.cu/mod/resource/view.php?id=33748.**

Falgueras, Benet Campderrich. 2003. ***Ingeniería de Software.*** **s.l. : Editorial UOC, 2003.**

—. 2003. ***Ingeniería de Software.*** **s.l. : Editorial UOC, 2003.**

Geeks & Linux Atelie! 2010. **Geeks & Linux Atelie! [En línea] 2010. [Citado el: 29 de 10 de 2010.] http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/.**

Hernández Becerra, Edgar. 1993. ***Monitoreo y evaluación de logros en proyectos de ordenación de cuencas hidrográficas.*** **Mérida : s.n., 1993.**

ILCE (Instituto Latinoamericano de la Comunicación Educativa). 2006. **Biblioteca Digital. [En línea] 2006. [Citado el: 26 de 11 de 2010.] http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen3/ciencia3/149/htm/sec_5.htm.**

Ivar Jacobson, Grady Booch y James Rumbaugh. 2000. ***El Proceso Unificado de Desarrollo de Software.*** **Madrid : Addison Wesley, 2000.**

Jiménez Lezama, Juan Carlos. 2005. ***"MODELADO DE UN BEAN DE BÚSQUEDA MEDIANTE UML".*** **2005.**

La Zona Linux. 2010. **La Zona Linux. [En línea] 2010. [Citado el: 28 de 10 de 2010.] http://lazonalinux.com.ar/post/281/como-programar-en-linux-primera-parte.html.**

Lanzillotta, Analía. 2009. **Mastermagazine. [En línea] 2009. http://www.mastermagazine.info/termino/5560.php.**

Lenguajes de Programación. 2010. **Lenguajes de Programación. [En línea] 2010. [Citado el: 23 de 11 de 2010.] http://www.lenguajes-de-programacion.com/programacion-java.shtml..**

López Meléndez, Elizabeth. 2009. ***BASE DE DATOS PARA EXPEDIENTES MÉDICOS.*** **Puebla : s.n., 2009.**

Márquez Avendaño, Bertha Mariel y Zulaica Rugarcía, José Manuel. 2004. ***Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invidentes Dos-Vox en español .*** **México : s.n., 2004.**

Masip, David. 2002. **desarrolloweb.com. [En línea] 19 de 7 de 2002. [Citado el: 15 de 1 de 2011.] http://www.desarrolloweb.com/articulos/840.php.**

Ministerio de Educación Perú. 2007. ***Orientaciones para el Monitoreo y Evaluación de los Planes Operativos de las Direcciones Regionales de Educación y Unidades de Gestión Educativa Local, dependientes de los Gobiernos Regionales.*** **Lima : s.n., 2007.**

Otero Vidal, Mari Carmen. 2003. ***Lenguajes y Sistemas Informáticos.*** **Donostia : s.n., 2003.**

Pecos, Daniel. 2002. **danielpecos.com. [En línea] 7 de 6 de 2002. [Citado el: 17 de 1 de 2011.] http://danielpecos.com/docs/mysql_postgres/index.html.**

PostgreSQL, Portal en español sobre. 2010. **Portal en español sobre PostgreSQL. [En línea] 2 de 10 de 2010. [Citado el: 8 de 12 de 2010.] http://www.postgresql-es.org/sobre_postgresql.**

Rivera López, Alejandro. 2008. ***Sistema asistente para la generación de horarios de cursos.*** **México : s.n., 2008.**

Schildt, Herbert. 1995. ***C++ guía de autoenseñanza.*** **España : s.n., 1995.**

sdceefve. 2010. ***dhrtjhytrj.*** **2010.**

Sistemas de Seguridad, Vigilancia y Monitoreo Remoto. **Sistemas de Seguridad, Vigilancia y Monitoreo Remoto. . [En línea] [Citado el: 4 de 11 de 2010.] http://www.gscssoftware.com/tecsoftmon.htm..**

Software, Colectivo de Autores Asignatura Ingeniería de. 2011. **Entorno Virtual de Aprendizaje. [En línea] 2011. [Citado el: 15 de 2 de 2011.] http://eva.uci.cu/mod/resource/view.php?id=35381.**

Sommerville, Ian. 2005. ***Ingeniería del Software.*** **2005.**

Stream Labs, Television computer systems. 2010. **Stream Labs, Television computer systems. [En línea] 2010. [Citado el: 24 de 11 de 2010.] http://www.streamlabs.es/products/devices/multiscreen/index.php..**

Tedial. 2009. **Tedial. [En línea] 2009. [Citado el: 20 de 11 de 2010.] http://www.tedial.com/index.php?option=com_content&view=article&id=46&Itemid=54.**

Vargas, Félix. ***Análisis y Diseño de Algoritmos.***

BIBLIOGRAFÍA CONSULTADA

Alonso Guerrero, Zorilin y Leyva González, Genry. 2009. ***Sistema de Captura y Transcripción de Audio.*** **La Habana : s.n., 2009.**

Beck, Kent. 1999. ***Beck, K. Extreme Programming Explained.Embrace Change.*** **1999.**

Calderón, Amaro, y otros. 2007. ***Metodologías Ágiles.*** **Trujillo Perú : s.n., 2007.**

Castañeda, Gerardo Gerardo. 2006. ***LOS ARCHIVOS AUDIOVISUALES EN LAS REDES DIGITALES DE COMUNICACIÓN PARA LA EDUCACIÓN Y LA CULTURA.*** **Madrid, España. : s.n., 2006. ISBN (PUBLICACIÓN EN CD-ROM): 84-369-4135-7.**

Cerezo López, Yolanda, Peñalba Rodríguez, Olga y Caballero Roldán, Rafael. 2007. ***Iniciación a la programación en C#: un enfoque práctico.*** **2007.**

Colectivo. 2008-2009. **Arquitectura y Patrones de diseño. 2008-2009.**

Corvo Roque, Serguey. 2010. ***Interfaz de comunicación con la librería libvlc para las aplicaciones de reproducción y transmisión de media dentro del departamento de Señales Digitales.*** **La Habana : s.n., 2010.**

Cruz Chávez, Marco Antonio. 2010. **Universidad Autónoma del Estado de Morelos. [En línea] 4 de 3 de 2010. [Citado el: 17 de 1 de 2011.] http://www.uaem.mx/posgrado/mcruz/cursos/miic/MySQL.pdf.**

Davis, William. 1992. ***Herramientas CASE: metodología estructurada para el desarrollo de los sistemas.*** **1992.**

Edmondson, Ray. 2004. ***Filosofía y principios de los archivos audiovisuales.*** **París, : s.n., 2004.**

Entorno Virtual de Aprendizaje. 2010. **Entorno Virtual de Aprendizaje. [En línea] 2010. [Citado el: 24 de 10 de 2010.] http://eva.uci.cu/mod/resource/view.php?id=33748.**

Estrada Velazco, Aylin y Suárez Pérez, Jean Michael. 2009. ***Sistema de Captura e Indexación de Video.*** **La Habana : s.n., 2009.**

Falgueras, Benet Campderrich. 2003. ***Ingeniería de Software.*** **s.l. : Editorial UOC, 2003.**

—. 2003. ***Ingeniería de Software.*** **s.l. : Editorial UOC, 2003.**

García, Córdova. 2004. ***La tesis y el trabajo de tesis.*** **México: Limusa : s.n., 2004.**

Geeks & Linux Atelie! 2010. **Geeks & Linux Atelie! [En línea] 2010. [Citado el: 29 de 10 de 2010.] http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/.**

Gómez, Laureano Felipe. 2009. ***MEMORIAS DE PRÁCTICAS EN EL USO DEL INDIZADOR SWISH-E .*** **2009.**

Gracia, Joaquin. 2005. **Patrones de diseño.Anális y Diseño. Ingeniería de Software. [En línea] 27 de 5 de 2005. http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.**

Hernández Alba, Yailien y l Duvier Fernández Pérez, Leonel Duvier Fernández Pérez. 2008. ***Guía para la gestión de riesgos a través de RUP.*** **La Habana : s.n., 2008.**

Hernández Becerra, Edgar. 1993. ***Monitoreo y evaluación de logros en proyectos de ordenación de cuencas hidrográficas.*** **Mérida : s.n., 1993.**

Hernández León, Rolando Alfredo y Coello González, Sayda. 2002. ***EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA.*** **La Habana : s.n., 2002.**

ILCE (Instituto Latinoamericano de la Comunicación Educativa). 2006. **Biblioteca Digital. [En línea] 2006. [Citado el: 26 de 11 de 2010.] http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen3/ciencia3/149/htm/sec_5.htm.**

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. ***El Proceso Unificado de Desarrollo de Software.*** **Madrid : s.n., 2000.**

Jiménez Lezama, Juan Carlos. 2005. ***"MODELADO DE UN BEAN DE BÚSQUEDA MEDIANTE UML".*** **2005.**

La Zona Linux. 2010. **La Zona Linux. [En línea] 2010. [Citado el: 28 de 10 de 2010.] http://lazonalinux.com.ar/post/281/como-programar-en-linux-primera-parte.html.**

Lanzillotta, Analía. 2009. **Mastermagazine. [En línea] 2009. http://www.mastermagazine.info/termino/5560.php.**
Larman, Craig. 1999. ***UML y Patrones Introducción al análisis y diseño orientado a objetos.*** **México : PRENTICE HALL, 1999.**
López Meléndez, Elizabeth. 2009. ***BASE DE DATOS PARA EXPEDIENTES MÉDICOS.*** **Puebla : s.n., 2009.**
Márquez Avendaño, Bertha Mariel y Zulaica Rugarcía, José Manuel. 2004. ***Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invidentes Dos-Vox en español .*** **México : s.n., 2004.**
Martínez Alvarez, Juan Pablo. 2008. ***"EL PRODUCT PLACEMENT O LA PUBLICIDAD ENCUBIERTA Y SU USO EN LOS MEDIOS AUDIOVISUALES".*** **Guatemala : s.n., 2008.**
—. 2008. ***"EL PRODUCT PLACEMENT O LA PUBLICIDAD ENCUBIERTA Y SU USO EN LOS MEDIOS AUDIOVISUALES".*** **Guatemala : s.n., 2008.**
Masip, David. 2002. **desarrolloweb.com. [En línea] 19 de 7 de 2002. [Citado el: 15 de 1 de 2011.] http://www.desarrolloweb.com/articulos/840.php.**
Mauricio, Beuchot. 2007. ***Introducción a las ciencias de la computación con JAVA.*** **2007.**
Molina, Mería Pinto. 2004. **Búsqueda y recuperación de Información. [En línea] 2004. [Citado el: 23 de noviembre de 2010.] http://www.mariapinto.es/e-coms/recu_infor.htm.**
Montiel, Patricia Marín. 2002. **excalibur. [En línea] 2002. [Citado el: 1 de octubre de 2010.] http://personales.upv.es/ccarrasc/doc/2001-2002/excalibur/excalibur.htm#_Toc9859546.**
Nevado Cabello, Ma Victoria. ***Introducción a Las Bases de Datos Relacionales.***
Otero Vidal, Mari Carmen. 2003. ***Lenguajes y Sistemas Informáticos.*** **Donostia : s.n., 2003.**
Pecos, Daniel. 2002. **danielpecos.com. [En línea] 7 de 6 de 2002. [Citado el: 17 de 1 de 2011.] http://danielpecos.com/docs/mysql_postgres/index.html.**
PostgreSQL, Portal en español sobre. 2010. **Portal en español sobre PostgreSQL. [En línea] 2 de 10 de 2010. [Citado el: 8 de 12 de 2010.] http://www.postgresql-es.org/sobre_postgresql.**
Pressman, Roger S. 2001. **Ingeniería de Software.** ***Ingeniería de Software. Un enfoque práctico.*** **s.l. : Mc Graw Hill, 2001.**
Pupo Gómez, Reynier. 2010. ***Gestor de ficheros para el Entorno de Escritorio.*** **La Habana : s.n., 2010.**
Ramón, Tania Cerezo. **Análisi de la herramienta Google Desktop. [En línea] [Citado el: 1 de octubre de 2010.] http://personales.upv.es/ccarrasc/doc/2004-2005/BuscadoresWeb/trabajosrpDEFINITIVO.htm#_Toc104899909.**
Raquel, Mariana. 2009. ***Análisis Cualitativo y Cuantitativo de Herramientas de Entorno Visual para Desarrollo Web en PHP aplicado a la EPEC.*** **Ecuador : s.n., 2009.**
Raul Yanquen, JORGE FORERO,ANDREA PEÑA. 2010. **Tecnología de Redes Universidad de la Salle Sistema de Información. [En línea] 19 de 5 de 2010. [Citado el: 15 de 2 de 2011.] http://tecnologiaorredes.blogspot.com/2010_05_19_archive.html.**
Rivera López, Alejandro. 2008. ***Sistema asistente para la generación de horarios de cursos.*** **México : s.n., 2008.**
Riveros, Héctor G y Rosas, Lucía. 1982. ***EL método científico aplicado a las ciencias experimentales.*** **México : s.n., 1982.**
Sanchez, María A. Mendoza. 2004. **[En línea] 2004. [Citado el: 25 de noviembre de 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.**
Semanat Aldana, Edmis Deivis y Verdecia Four, Leonor. 2009. ***Sistema de Video Vigilancia.*** **LA Habana : s.n., 2009.**
Serrano Rosales, Carlos Luis y Rodríguez Vázquez, Solangel. 2008. ***Desarrollo de Biblioteca de Métodos Numéricos (BMN), referente a Sistemas de Ecuaciones Lineales, Sistemas de Gran Dimensión y Poco Densos e Integración Numérica.*** **La Habana : s.n., 2008.**

Sistemas de Seguridad, Vigilancia y Monitoreo Remoto. **Sistemas de Seguridad, Vigilancia y Monitoreo Remoto. . [En línea] [Citado el: 4 de 11 de 2010.] http://www.gscssoftware.com/tecsoftmon.htm..**
Software, Colectivo de Autores Asignatura Ingeniería de. 2011. **Entorno Virtual de Aprendizaje. [En línea] 2011. [Citado el: 15 de 2 de 2011.] http://eva.uci.cu/mod/resource/view.php?id=35381.**
Sommerville, Ian. 2005. ***Ingeniería del Software.* 2005.**
Tedial. 2009. **Tedial. [En línea] 2009. [Citado el: 20 de 11 de 2010.] http://www.tedial.com/index.php?option=com_content&view=article&id=46&Itemid=54.**
Torres, Patricio Letelier. 2003. **[En línea] 2003. [Citado el: 15 de noviembre de 2010.] http://oness.sourceforge.net/proyecto/html/ch05s02.html.**
Vargas, Félix. ***Análisis y Diseño de Algoritmos.***

GLOSARIO DE TÉRMINOS

Media: Película, imagen o cualquier otro material audio visual que requiere de un uso especial de equipamiento para visualizarlo.

Audiovisual: se refiere a la combinación de lo visual y lo auditivo, a todas las imágenes acompañadas de sonidos.

Librería: Conjunto de subprogramas utilizados para desarrollar software.

UML: “Unified Modelling Language”, lenguaje de modelado gráfico que permite especificar, construir, visualizar y documentar los artefactos de un sistema utilizando el enfoque orientado a objetos.

CASE: Ingeniería de Software Asistida por Computación (CASE).Se define como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación.

SGBD: Sistemas gestores de base de datos, tipo de software que se utilizan para servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

IDE: Entorno integrado de desarrollo

Framework: Es un conjunto de componentes que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas WEB.

Interfaz: Zona de contacto o conexión entre dos componentes de "hardware"; entre dos aplicaciones, o entre un usuario y una aplicación.