UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

Título: Componente para el Procesamiento Digital de Audio en Sistemas de Información Audiovisual.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor

Janet Cristina Labrada Paneque

Tutor

MSc. Ruber Hernández García

Cotutor

José Andrés Hernández Bustio

Ciudad de la Habana, junio de 2012

"Año de 54 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor del trabajo **"Componente para el Procesamiento Digital de Audio en Sistemas de Información Audiovisual"** y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

Autor: Janet Cristina Labrada Paneque **Tutor:** MSc. Ruber Hernández García

DATOS DE CONTACTO

Tutor: MSc. Ruber Hernández García

Formación académica:

Ingeniero en Ciencias Informáticas, Universidad de Ciencias Informáticas, 2007.

Centro Laboral:

Centro de Desarrollo de Geoinformática y Señales Digitales. Facultad 6.

Correo electrónico: rhernandezg@uci.cu

Cotutor: Ing. José Andrés Hernández Bustio

Formación académica:

Ingeniero en Ciencias Informáticas, Universidad de Ciencias Informáticas, 2008.

Centro Laboral:

Centro de Desarrollo de Geoinformática y Señales Digitales. Facultad 6.

Correo electrónico: jahernandez@uci.cu

RESUMEN

Los sistemas de información audiovisual revolucionan el mundo actual por la magnitud de información y conocimientos que difunden. Como parte de la información que brindan se encuentran los archivos de audio que juegan un papel fundamental en la comunicación por vía directa a los seres humanos. Aunque aparentemente la utilización de audios no tiene mayor dificultad es posible encontrarse con sistemas de información que necesitan realizar un procesamiento digital sobre este tipo de archivo para lograr obtener un resultado favorable cuando se reproduzcan. El presente trabajo de diploma contiene la investigación y el proceso de desarrollo realizado para obtener un componente de procesamiento digital de audio que permita eliminar la baja calidad, escasa estandarización y duplicado de los archivos de audios existentes en los sistemas de información audiovisual del centro de desarrollo Geoinformática y Señales Digitales. Con el fin de obtener los mejores resultados el desarrollo se realiza bajo la guía de la metodología XP, la aplicación es implementada haciéndose uso del lenguaje de programación C++ y el framework de desarrollo Qt. Se presentan los requisitos identificados y el cumplimiento de los mismos en la aplicación final, que es validada además por las pruebas realizadas propuestas por la metodología de desarrollo utilizada. El impacto social de los resultados de esta investigación se centra en un componente para el procesamiento digital de audio que solucione los problemas presentes en los sistemas de información.

Palabras claves: audio, componente, procesamiento, sistemas.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El avance de la ciencia y el desarrollo de la tecnología han propiciado que la utilización de archivos multimedia constituya en la actualidad una vía de adquisición del conocimiento para los seres humanos. La posibilidad de interactuar con archivos de medias (audio, imagen y video) conlleva a un mayor entendimiento de la información que se maneja de forma digital. Las sociedades contemporáneas emplean los medios audiovisuales como una mejor forma de comunicación debido a la información que mediante ellos se puede expresar, además de ser el principal recurso de entretenimiento desde edades muy tempranas.

El arte de comunicar ha crecido y cobrado fuerza debido a la importancia que encierra, por lo que muchos autores han dedicado sus estudios a profundizar en la mejor forma de utilizarlo y propagarlo. Relacionado a esto Robert Kiyosaki planteó:

> *"La capacidad de vender, de comunicarse con otro ser humano, cliente, empleado, jefe, esposa o hijo, constituye la base del éxito personal. Las habilidades de comunicación como escribir, hablar y negociar son fundamentales para una vida exitosa"* (Kiyosaki, y otros, 1997).

El auge alcanzado por el uso de los medios audiovisuales ha propiciado el surgimiento de sistemas de información orientados al tratamiento y administración de los archivos multimedia. Estos sistemas almacenan todo tipo de información y se difunden en diferentes formatos. Su surgimiento ha optimizado procesos básicos y necesarios para la vida de los hombres como son la comunicación oral y la relación directa entre los seres humanos, convirtiéndose en el canal para nutrir a las personas de las respuestas a todas sus interrogantes y de un gran volumen de información que se encontraba anteriormente fuera de su alcance.

Con el fin de lograr una mayor difusión de la información en Cuba se utilizan sistemas audiovisuales para alcanzar este propósito, con la premisa de elevar la cultura del pueblo, así como la total claridad sobre la realidad del país y de otros países del mundo. La Universidad de las Ciencias Informáticas (UCI) representa un papel fundamental en las futuras proyecciones del país. Surgida al calor de la batalla de idea como un sueño del Comandante en Jefe Fidel Castro Ruz, denominado "Proyecto Futuro", con dos objetivos fundamentales: informatizar el país y desarrollar la industria del software para contribuir al desarrollo económico del mismo. El centro de desarrollo Geoinformática y Señales Digitales (GEYSED), perteneciente a la facultad 6 de dicha universidad, cuenta con el departamento Señales Digitales donde se

desarrollan sistemas de procesamiento y transmisión de audiovisuales convirtiéndose en productos valiosos para la informatización del país.

La utilización de elementos multimedia supone tres partes fundamentales que conforman las medias: el audio, la imagen y el video, siendo importantes los tres para la correcta administración de este tipo de archivo. El audio es una señal eléctrica que se representa en forma de onda y se propaga dentro del rango de frecuencias perceptibles por las personas. El desarrollo y evolución del mismo dio origen a los formatos de audios que son un contenedor donde se guardan las grabaciones de audio. La diferencia entre estos tipos de archivo consiste en la forma en que se almacenan los datos y sus capacidades de reproducir la información.

El procesamiento digital consiste en un conjunto de técnicas que se aplican a las señales con el objetivo de mejorar su calidad y propagación en el medio. Sus aplicaciones se han multiplicado y se extienden a sectores tan necesarios como la medicina, la educación y las investigaciones científicas por citar algunos ejemplos. El procesamiento digital de audio se encarga específicamente de las señales de audio. Durante el procesamiento la señal se codifica y descodifica para su posterior reproducción, en este proceso se obtiene una señal digital libre de ruidos e interferencias a diferencia de las señales analógicas, además de editar sus parámetros, comprimirla, almacenarla, reconstruirla, amplificarla entre las múltiples funciones que se pueden realizar sobre ella durante el procesamiento.

En el departamento Señales Digitales existen proyectos reales que se encargan específicamente de la gestión de archivos de medias a través de canales de difusión de contenidos para los sistemas de información audiovisual. Como ejemplo de este tipo de aplicación se pueden mencionar: Plataforma de Televisión Informativa PRIMICIA, Sistema de Gestión y Transmisión de Contenidos Audiovisuales SIAV, Sistema de Captura y Catalogación de Medias SCCM, por mencionar algunos ejemplos.

PRIMICIA se encarga de la transmisión constante y automática de información mediante un canal televisivo. Tiene en su composición dos subsistemas: Administración que se basa en la gestión de toda la información del sistema y Transmisión que tiene a su cargo la visualización de las noticias y materiales publicados en la plataforma. SCCM desarrolla un sistema para la gestión de los procesos a los materiales audiovisuales. Entre las funciones que desarrolla se encuentra la gestión de medias, con acciones específicas como la codificación y catalogación sobre estos archivos. SIAV realiza la publicación y transmisión de contenidos multimedia mediante la web. Gestiona los contenidos multimedia, transmite,

monitoriza y gestiona los errores. Es una plataforma con varios subsistemas muy fáciles de personalizar a conveniencia de los clientes.

La gestión de las medias en el departamento Señales Digitales ha demostrado que existen problemas en el procesamiento digital de los archivos de audio en los sistemas desarrollados (Cruz, 2011). En los sistemas de información que lo utilizan se observa que la gran variedad de formatos de audios que interactúan en el mismo entorno, provoca que los archivos no estén bajo un estándar común y la calidad no sea óptima en muchos casos. Otro problema presentado es el duplicado de los archivos, no existe control en la entrada del audio al sistema por lo que las bases de datos almacenan elementos idénticos que no se detectan. Por estas razones se hace necesario establecer una estrategia para garantizar un procesamiento digital de audio satisfactorio que brinde la posibilidad de adoptar nuevas funcionalidades en el tratamiento de los audios para los sistemas de información.

Como una primera aproximación a la solución del problema se desarrolló un flujo de procesamiento digital de audio para garantizar la calidad de los archivos de audio que se utilizan en la Plataforma de Televisión Informativa PRIMICIA (Cruz, 2011). El procedimiento consiste en someter la señal de audio que entra a la plataforma a tres fases, en las cuales se verificará el formato del audio, del cual se confeccionará una huella digital que lo identificará unívocamente. Como último paso se efectuará el reconocimiento para comprobar si el archivo se encuentra duplicado en la base de datos. Uno de los principales problemas que presenta esta propuesta de solución es que se realizó para un producto en específico sin tener en cuenta los sistemas de información audiovisual de manera general.

Ante la problemática planteada se deriva el siguiente **problema a resolver**: La baja calidad, escasa estandarización y duplicado de los archivos de audio que se utilizan en los sistemas de información audiovisual del centro de desarrollo de Geoinformática y Señales Digitales provocan problemas en la reproducción de estos durante su transmisión.

A partir del problema científico se ha centrado el estudio de la solución en los componentes para el procesamiento audiovisual en sistemas de información lo cual constituye el **objeto de estudio**. Para resolver dicho problema se ha identificado como **objetivo general** de esta investigación desarrollar un componente de audio para los sistemas de información audiovisual del centro de desarrollo de Geoinformática y Señales Digitales que garantice la calidad, estandarización y eliminación de los archivos duplicados a través de algoritmos y técnicas de procesamiento automático de audio, con el fin de lograr una correcta reproducción y almacenamiento de los mismos.

Al haber planteado los elementos fundamentales que describen el problema a resolver y los objetivos sobre el cual se le dará solución se puede determinar que el **campo de acción** tendrá su incidencia en el componente para el procesamiento de audio en los sistemas de información audiovisual del departamento Sistemas Digitales del centro de desarrollo GEYSED.

Luego de obtener toda la información relacionada con la problemática que se estudia se plantéa la siguiente **idea a defender**: Si se implementa un componente para el procesamiento digital de audio en los sistemas de información audiovisual del departamento Sistemas Digitales del centro de desarrollo Geoinformática y Señales Digitales se garantizará la eliminación de los archivos duplicados, la estandarización y calidad de los formatos de audio, con el fin de lograr una correcta transmisión del audio.

Para lograr el objetivo general se tendrán en cuenta las siguientes **tareas de la investigación**:

1. Elaboración del marco teórico sobre componentes de procesamiento de audio.
2. Definición de las herramientas y el lenguaje de programación a utilizar.
3. Modelado del componente para el procesamiento del audio.
4. Implementación del componente para el procesamiento del audio
5. Validación del componente desarrollado.

El desarrollo de la investigación está dirigido mediante métodos científicos que se aplicaron a lo largo de la misma. El método científico “se basa en la observación sistemática de la realidad, su medición, el análisis de sus propiedades y características, la elaboración de hipótesis, su interpretación y contrastación, la formulación de alternativas de acción o respuesta, para producir ciencia” (León, y otros, 2011).

Métodos teóricos:

- Analítico – Sintético: Su utilización se ve reflejada en el estudio de la documentación especializada sobre los elementos que conforman los componentes para el procesamiento audiovisual en sistemas de información y la posibilidad de arribar a conclusiones al finalizar el desarrollo de la investigación.
- Análisis histórico lógico: Tiene su influencia en la valoración y análisis de algoritmos de detección de duplicados audiovisuales, técnicas de codificación digital y soluciones existentes

sobre manipulación de audios que inciden en el correcto desarrollo del producto deseado y en el cumplimiento del objetivo general de la investigación.

- Modelación: Este método es utilizado en la investigación para entender cómo funciona el flujo del procesamiento digital de audio en los sistemas de información audiovisual.

Métodos empíricos:

- Experimental: Al finalizar el desarrollo de la solución se valida el producto resultante, que en este caso se refiere a un componente de audio para sistemas de información audiovisual.

Estructura del Documento

El documento se encuentra dividido en tres capítulos. En el primero, Fundamentación Teórica, se realiza la elaboración del marco teórico sobre componentes de procesamiento de audio. Además, se efectúa una evaluación de los resultados obtenidos en el trabajo precedente acerca de los algoritmos, técnicas y soluciones existentes para eliminar el duplicado de los audios, mejorar su calidad y estandarización. De igual manera se definen las herramientas y el lenguaje de programación a utilizar, así como la metodología de desarrollo más recomendable.

En el segundo capítulo, Presentación de la Solución Propuesta, se modela el componente, se identifican los requisitos, se diseña la arquitectura, las clases, y para finalizar se realiza la implementación del componente para el tratamiento del audio en sistemas de información audiovisual.

En el tercer capítulo se valida el componente desarrollado. Además se realizan las pruebas de integración al componente, donde se muestra el resultado de las pruebas realizadas para demostrar que la solución es correcta. Finalmente se dan las conclusiones generales y un conjunto de recomendaciones con vistas a trabajos futuros.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Este capítulo constituye la base teórica para avanzar en el desarrollo de la problemática existente y el cumplimiento de los objetivos planteados. Para su confección se realiza un profundo análisis de los elementos que conforman el objeto de estudio de la investigación, con la explicación de cada uno de ellos en cuanto a las aplicaciones prácticas que poseen, dando la posibilidad de comprender el marco de estudio donde se sustentan.

1.1 Elementos asociados al dominio del problema

Cuando se realiza una investigación nunca se debe comenzar por lo concreto, sino que primero se efectúa una búsqueda de la representación abstracta contenida en el conocimiento precedente. En esta búsqueda de información surgen conceptos y definiciones relacionados con el objetivo que dirige el proceso de desarrollo que se lleva a cabo. Para una mejor comprensión de la situación problemática se relacionan en lo adelante los principales conceptos que están asociados al desarrollo de la investigación.

1.1.1 Procesamiento digital de audio

El procesamiento digital tiene su fundamento en el análisis y la interpretación de señales de cualquier tipo, para su posterior reproducción con el uso de herramientas y técnicas que lo posibiliten. El procesamiento digital de audio se deriva del procesamiento digital de señales y se especializa en el tratamiento de la señal de audio. El audio constituye una fuente de transmisión de información que es codificada para poder difundirse, y tiene como ventaja que puede ser almacenado para una posterior recuperación y utilización. La presente investigación centrará el enfoque del procesamiento digital de audio en la adición y corrección, supresión de duplicados y validación de la calidad de los audios en los sistemas de información audiovisuales.

Calidad de los formatos de audio

Un formato de audio como contenedor multimedia guarda una grabación de audio en los niveles de frecuencia audible para los seres humanos. Los formatos de audio poseen parámetros definidos entre los que se pueden citar la frecuencia de muestreo, el número de canales y los números de bit por muestra.

Cuando se habla de estandarización se refiere a establecer una igualdad en los archivos de audio que se encuentran almacenados en los sistemas audiovisuales, de forma tal que la diferencia entre las características de los formatos no sea un impedimento para el correcto funcionamiento del mismo. En los sistemas de información del departamento Señales Digitales se hace necesario aplicar un control de la entrada de los formatos de audio a la plataforma, para que al estar almacenados todos cuenten con la calidad requerida por el sistema para su posterior reproducción.

Con el objetivo de que los archivos de audio cuenten con la calidad óptima para el sistema, en este mismo trabajo, se establecieron valores medibles para cada parámetro que componen los audios. Para la asignación de los valores de calidad también fue necesario recurrir a las particularidades de los formatos de audios predeterminados para la interacción con el sistema. Todo el análisis concluyó en que: la frecuencia de muestreo sea 44.1 KHz; la tasa de bits sea variable; y el número de canales igual a 2 (estéreo) (Cruz, 2011).

Según el flujo propuesto por la autora de referencia (*Ver Anexo # 1*), cuando un audio entra al sistema pasa por un proceso de extracción de parámetros para comprobar la calidad que presenta, mediante los principales parámetros que tienen asociados los formatos de audio. Luego se comprueban los parámetros de calidad definidos en el patrón de calidad del sistema.

De ser positivo la señal de audio pasa a una fase superior del procesamiento, sino se analiza si la frecuencia de muestreo está por encima de 32 KHz, siendo esta la frecuencia de muestreo mínima que necesita un archivo para tener calidad, de no ser así se le informa al usuario que el archivo no tiene calidad para ser procesado, así finaliza el proceso, si es mayor se le cambian los parámetros al archivo de audio en correspondencia con lo definido en el patrón de calidad (Cruz, 2011).

Eliminación de duplicados

El duplicado de los audios es actualmente uno de los principales problemas que presentan los sistemas de información audiovisual del departamento Señales Digitales. Para darle solución se propuso una técnica que utiliza la huella digital de audios con el objetivo de detectar si el archivo se encontraba en el sistema. Cada huella digital es única, lo que quiere decir que un archivo tendrá una propia, de estar repetida indica que el archivo esta duplicado (Cruz, 2011).

Una huella digital acústica robusta estará basada en las características perceptivas del audio. Si dos archivos suenan idénticos a un oyente humano, aunque sus datos binarios diferencien, sus huellas digitales acústicas deben emparejar (Allamanche, y otros, 2001).

Para obtener una huella digital primeramente se deben extraer las características del archivo de audio y con esos elementos conformar la huella digital de cada audio. Existen diferentes métodos para la extracción de las características propias de un archivo de sonido, entre los que se pueden mencionar: Coeficientes Cepstrales de Frecuencia Mel (MFCC) y Vectores de Croma (VC).

MFCC compara parejas de fragmentos de un segundo de duración y evalúa la similitud en el timbre. Las bandas de frecuencia que analiza están espaciadas según la escala de Mel que es una representación de la percepción humana de la altura. Dichas bandas cubren las frecuencias de 20Hz a 3 KHz. El vector de características se compone de 12 coeficientes (Saráchaga, y otros, 2006).

VC compara parejas de fragmentos de un segundo de duración, pero a diferencia de MFCC, analiza el contenido melódico (semitonos) en lugar del timbre. Analiza frecuencias fundamentalmente entre 130 Hz y 4 KHz. Se consideran los componentes de cada croma (12 coeficientes) sin distinguir entre las distintas octavas (Saráchaga, y otros, 2006).

Al analizar las diferencias entre ambos métodos, (Cruz, 2011) propuso el método Coeficientes Cepstrales de Frecuencia Mel como algoritmo de extracción de características a aplicar en el procesamiento digital de audio en los sistemas de información audiovisual del departamento Señales Digitales.

Después de haber pasado por las fases de extracción de las características y ajuste al patrón de calidad se procede a la extracción de la huella digital. Se realizan transformaciones sobre la señal de audio dividiéndola en bloques sin perder ningún fragmento de sonido. Al utilizar el algoritmo MFCC se obtienen una secuencia de vectores de parámetros, que cumplen dos objetivos simultáneamente: representar las características esenciales de la señal de voz, así facilitan su reconocimiento y reducen la cantidad de información (Maldonado, 2003).

Para el modelado de la huella digital de audio, (Cruz, 2011) propone utilizar dos técnicas para fortalecer el reconocimiento de la huella. La primera técnica es la normalización de la media cepstral que consiste en calcular el valor medio de cada coeficiente cepstral en el audio seleccionado y normalizarlo con la substracción de ese valor medio. Permite modificar las estadísticas de las características de voz ruidosa y hacerlas coincidir con las de una referencia limpia (Martínez, 2007). La segunda técnica utilizada es la

varianza o media cuadrática de las puntuaciones diferenciales. Para hallarla, se resta a cada valor la media y se eleva al cuadrado; se repite para todos los valores, se suman todos los cuadrados y se divide por el número de valores (Rosales, 2010).

La fase final del flujo lo constituye el reconocimiento donde ya se cuenta con la huella digital de audio del archivo que se analiza en el momento de la interacción de un audio entrante al sistema. Esta huella se compara con las demás huellas que están en la base de datos de huellas digitales pertenecientes a los audios almacenados en la base de datos de media. Este proceso consume una gran cantidad de tiempo por la complejidad de su aplicación. El flujo de procesamiento termina cuando el sistema le ofrece al usuario una respuesta de si el audio se encontraba o no en el sistema después de haber comparado con cada una de las huellas existentes en la base de datos (*Ver Anexo # 1*), (Cruz, 2011).

1.1.2 Sistemas de información audiovisual

Un sistema de información (SI) es un conjunto organizado de elementos, que pueden ser personas, datos, actividades o recursos materiales en general. Estos elementos interactúan entre sí para procesar información y distribuirla de manera adecuada en función de los objetivos de una organización (Briceño, 2008).

Para un funcionamiento óptimo, en los sistemas de información deben ocurrir cuatro procesos fundamentales dirigidos al tratamiento de la información que se gestiona:

- la entrada de la información, donde se obtienen los datos mediante dispositivos de almacenamiento o captura;
- el almacenamiento, que posibilita la recuperación de información anteriormente guardada para su posterior empleo;
- el procesamiento, que transforma los datos originales en información utilizable en diferentes procesos;
- la salida de la información, que posibilita el flujo de la información al exterior de un dispositivo o sistema.

De manera general los SI se utilizan para obtener, almacenar, manipular, administrar, mover, controlar, desplegar, intercambiar, transmitir y recibir datos que tendrán una función comunicativa en el medio donde se muestre.

Los sistemas de información audiovisual son un tipo de SI con dos componentes relacionados, el elemento visual y el auditivo, que integrados ofrecen un medio distinto de comunicación. Para una mejor gestión de la información audiovisual se hace necesario poseer cualidades como la interactividad, que brinda al usuario una interacción de manera activa y dinámica con el medio; la iconicidad, donde se representa siempre la realidad, corresponde al grado de realismo de una imagen con respecto al objeto que representa, el lado contrario de la iconicidad es la abstracción, donde la realidad es despojada de elementos reduciéndolos a categorías mentales; la sincronía, permite una comunicación directa y en tiempo real entre el emisor y el receptor, siendo la asincrónica una comunicación indirecta efectuándose en espacios de tiempo diferentes (Moles, 1982).

Gestión de medias en los sistemas de información audiovisual

Dentro de un SI se realizan acciones como la gestión y la catalogación de medias para ubicar o almacenar cada recurso audiovisual según criterios definidos, como puede ser el tipo de archivo. Al ser almacenados se puede operar sobre ellos con funciones como eliminar, modificar, visualizar, recuperar entre muchas otras. Para su correcta aplicación se necesita adecuar el entorno de trabajo a las condiciones existentes, almacenar organizadamente los archivos audiovisuales y controlar toda operación que se realice sobre ellos. Para el almacenamiento, se debe usar un servidor de ficheros para las medias y una referencia en base de datos para luego localizarlas, las cuales tienen criterios de búsqueda bien definidos y en las que se guardan varios datos de cada fichero, con el fin de lograr acceder al recurso de forma sencilla y práctica (Milián, 2011).

Para el tratamiento del audio en los SI audiovisual las acciones se realizan como plantéa (Milián, 2011). La entrada de la información al sistema es el proceso donde se obtienen los datos que se necesitan procesar; se realiza de forma manual, es el usuario quien proporciona de forma directa la información al sistema. Los audios se obtienen tanto de dispositivos externos como a través de la web. La salida de la información consiste en sacar al exterior el contenido del audio, es decir, reproducirlo para que transmita la información que posee.

1.1.3 Componentes de software

El concepto de componentes de software ha cobrado fuerza debido a la importancia de su implementación sobre un producto determinado. Los mismos son una parte modular en un sistema que facilita y brinda un conjunto de funcionalidades definidas. Su utilización posibilita que sus funciones se puedan utilizar en varios sistemas sin necesidad de repetir el código una vez más, por eso entre sus principales características se encuentran el ser reutilizable, intercambiable, cohesivos, lo que facilita su manejo y desarrollo dentro de un entorno colaborativo. La necesidad de estandarizar los elementos de un producto trajo consigo el auge de los componentes de software, comenzar un proyecto desde cero es una tarea bastante difícil e innecesaria si se contara con una herramienta que facilite su realización.

Al surgir la programación orientada a objetos se produce un cambio de paradigma muy grande en el desarrollo de software, donde se pasó de la descripción algorítmica de la solución a la representación y manipulación de los objetos que forman parte del problema. Esta evolución en la programación abrió las puertas a nuevas formas de desarrollo como la basada en la reutilización de componentes mediante la creación de componentes genéricos, fáciles de integrar, distribuidos e independientes de las plataformas de desarrollo (Montilva, y otros, 2003).

Características de los componentes de software

Existen algunos requisitos indispensables que deben cumplir los sistemas para que puedan ser catalogados como componentes de software (Rojas, y otros, 2004):

- Identificable: Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.
- Auto contenido: Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.
- Puede ser remplazado por otro componente: Se puede remplazar por nuevas versiones u otro componente que lo remplace y mejore.
- Con acceso solamente a través de su interfaz: Debe asegurar que estas no cambiaran a lo largo de su implementación.
- Escalable: Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.

- Bien documentado: Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, etc.
- Es genérico: Sus servicios debe servir para varias aplicaciones.
- Reutilizado dinámicamente: Puede ser cargado en tiempo de ejecución en una aplicación.
- Independiente de la plataforma: Debe funcionar con el uso de cualquier hardware, software o sistema operativo.
- Mantenido: Es deseable que esté inmerso en un proceso de mejoramiento continuo.
- Certificado: Debe estar certificado por una agencia de software independiente o mediante la aplicación de modelos de auto-certificación.
- Accedido uniformemente sin importar su localidad: La forma de invocar los servicios debe ser independiente de su ubicación (local o remota).

Reutilización de software

En el trabajo "Desarrollo de software basado en componentes", realizado por (Montilva, y otros, 2003), se plantea:

> *"La reutilización de software es un proceso de la Ingeniería de Software que conlleva al uso recurrente de activos de software en la especificación, análisis, diseño, implementación y pruebas de una aplicación o sistema de software".*

La reutilización de componentes de software constituye una opción muy favorable para la implementación de sistemas. En estudios realizados se ha demostrado que entre el 40% al 60% aproximadamente del código fuente de un software es reutilizable por otra aplicación con elementos semejantes. De igual forma un 75% de las funciones son comunes para varios sistemas, con un 15% de código único y novedoso que no se debe reutilizar por la particularidad que posee con el sistema al que pertenece. Al reutilizar código en diferentes aplicaciones se gana en tiempo, se ahorran recursos y esfuerzo, como elementos vitales para el desarrollo que propician además que se incremente la calidad del software implementado, como el tiempo de desarrollo es menor la productividad aumenta y los posibles riesgos se hacen menores (Montilva, y otros, 2003).

Clasificación de componentes de software

La reutilización e infraestructura de componentes de la ingeniería de software de negocio es una clasificación que busca definir las diferencias claves existentes entre los componentes, su complejidad, alcance, nivel de funcionalidad, etc.

Basado en estos principios (Heineman, y otros, 2001) establecen la siguiente clasificación:

1. Componentes GUI (interfaz gráfica de usuario): son aquellos que permiten construir interfaces para el usuario. La construcción de un componente GUI robusto exige por parte del desarrollador un alto nivel de conocimiento y compromiso con la ingeniería de software basada en componentes.
2. Componentes de Servicio: Este tipo de componentes proveen servicios comunes requeridos por diferentes aplicaciones, como acceso a bases de datos, servicios de mensajería y transacciones e integración con servicios del sistema. Una de las principales características de estos componentes, es que todos necesitan usar sistemas o infraestructura adicional para poder ejecutar su funcionalidad.
3. Componentes de Dominio: Estos componentes, cuando son reusables, pertenecen sólo a un dominio de aplicación específico, y no pueden ser reutilizados fuera de él. Estos componentes son difíciles de diseñar y construir. Por ser parte de un dominio de aplicación en particular, deben contar con una infraestructura de aplicación propia. El encargado del diseño e implementación del componente requiere un alto grado de conocimiento y experiencia dentro del dominio. Los casos de negocio que requieran dentro de su implementación componentes de dominio, requerirán tanto componentes GUI como de Servicio.

Estructuras que funcionan como tipos de componentes de software:

Para definir los tipos de componentes que existen hay que centrarse en la función que realizan y como varias de las estructuras existentes, antes de comenzar a utilizar el término componente, ya cumplían las funciones para las que está pensada la aplicación de esta tecnología. A consideración del autor de esta investigación las estructuras de software existentes que se comportan como componentes son las siguientes:

- Framework: es una aplicación incompleta y genérica, sobre la cual las aplicaciones que se implementan son las que le dan el estado final con la definición de una funcionalidad específica y propia. Para utilizar un framework es necesario la utilización y condensación de conocimientos para que este pueda ser utilizado como un componente.
- Plugins: es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API (Interfaz de programación de aplicaciones).
- Library: es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, donde pasan a formar parte de estos. Esto permite que el código y los datos se compartan y puedan modificarse.

Integración de un componente a una aplicación

Muchas de las aplicaciones que surgen en la actualidad para brindar servicios a las personas se construyen mediante la integración o composición de componentes de software. La integración que se produce puede graficarse como una interacción entre un cliente y un usuario (Ver Figura 3):

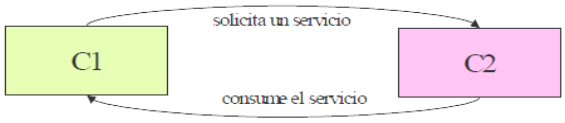

Figura 1: Interacción entre componentes.

El componente que hace la función de cliente realiza una solicitud al componente servidor, donde se satisface la solicitud efectuada y se devuelven los resultados al cliente.

No solamente los componentes se integran a los sistemas, sino que también los frameworks lo hacen. Existen tres clases de interacción en los sistemas basados en componentes (Montilva, y otros, 2003):

1. Componente-Componente (C-C): permite la interacción entre componentes. De este tipo de interacción se obtiene la funcionalidad de la aplicación, de forma tal que los contratos que especifican este tipo de interacción pueden ser clasificados como Contratos a Nivel de Aplicación.

2. Componente-*Framework* (C-F): posibilita las interacciones entre el *framework* y sus componentes. Dicha interacción permite que el *framework* administre los recursos de los componentes. Los contratos que especifican estas interacciones pueden ser clasificados como Contratos a Nivel de Sistema.

3. *Framework-Framework* (F-F): posibilita las interacciones entre *framewoks* y permiten la composición de componentes desplegados en *framewoks* heterogéneos. Estos contratos puede ser clasificados como Contratos de Interoperabilidad.

Proceso de desarrollo de software basado en componentes

(Sommerville, 2000) clasifica los procesos de desarrollo de software basados en la reutilización de componentes en dos categorías: desarrollo de componentes, que implica la adaptación o desarrollo de componentes con el propósito expreso de ser reutilizados en futuras aplicaciones. Su objetivo es producir repositorios de activos que puedan ser reutilizados en el desarrollo de software; desarrollo de software con reutilización de componentes, este enfoque maximiza la reutilización de componentes de software existentes y reduce el número de componentes que requieren ser desarrollados en su totalidad (desde cero). Para ser exitoso, este proceso demanda dos condiciones mínimas: la existencia de repositorios o bases de componentes reutilizables y que los componentes sean confiables y actúen de acuerdo a sus especificaciones.

Por otra parte Sametinger plantea una estrategia mucho más completa denominado Ciclo de vida gemelo, que divide el proceso en dos bloques, uno de Ingeniería de Dominios y el otro Ingeniería de Aplicaciones (Sametinger, 1997). Este modelo tiene como fin la reutilización de componentes de software a través de Ingeniería de Dominios (Ver Figura 4).

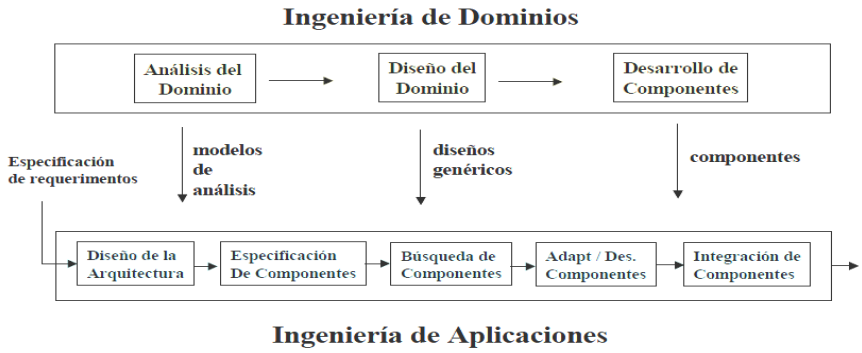

Figura 2: Modelo de procesos gemelos para el desarrollo de software basado en componentes.

1.2 Caracterización de la situación problemática

En el departamento Señales Digitales del centro de desarrollo GEYSED se desarrollan sistemas que gestionan información audiovisual. Ejemplo de ello lo constituyen los siguientes proyectos: Plataforma de Televisión Informativa PRIMICIA, Sistema de Gestión y Transmisión de Contenidos Audiovisuales SIAV, Sistema de Captura y Catalogación de Medias SCCM, por citar algunos. Estas plataformas procesan el contenido multimedia sin tener un control establecido para el tratamiento de estos archivos que incluye: eliminarlas, adicionarlas, modificarlas y reproducirlas en el sistema.

La plataforma PRIMICIA se encuentra estructurada en dos subsistemas que están estrechamente relacionados para brindar un resultado final eficiente y acorde a las necesidades de los usuarios. El subsistema de Transmisión es el que se encarga de la visualización de las noticias y materiales publicados en la plataforma, mientras que el subsistema de Administración realiza la gestión de las noticias y de los recursos multimedia que se utilizan para la elaboración de las mismas. El proceso de gestión de medias de la plataforma brinda la posibilidad de eliminarlas del sistema, adicionar sus datos, reproducirlas y modificarlas.

El proyecto SCCM ha direccionado sus funciones para en un futuro próximo posibilitar la gestión de fallas durante la digitalización de materiales audiovisuales, la asignación de tareas de edición y catalogación, la gestión de permisos para acceder a las medias, la gestión de las solicitudes y préstamos de materiales, la codificación de videos a varios formatos, la catalogación de medias, entre otras. El sistema está concebido sobre la filosofía de desarrollo basado en componentes. Cuenta con cuatro aplicaciones o módulos clientes: Catalogación, Control de Ingestas, Recuperación y Préstamos, y Administración; estos están encaminados a facilitar la interacción de los usuarios de la aplicación con el sistema.

Por su parte SIAV es un sistema capaz de automatizar procesos que se ejecutan en una entidad dedicada a la gestión, procesamiento y transmisión de contenidos multimedia. El sistema constituye una plataforma compuesta por varios subsistemas que pueden desplegarse independientes o integrados en un mismo sistema. La plataforma posee un subsistema web para la publicación y transmisión de contenido multimedia bajo demanda en la web. Además posee funcionalidades para realizar la programación de transmisiones de contenido multimedia, las transmisiones previamente programadas y la monitorización de las mismas, con la generación de reportes y otras funciones.

En los SI del departamento Señales Digitales se almacenan en una base de datos las operaciones que se pueden realizar sobre las medias. En el caso del audio, cuando se va a adicionar en el sistema se deben entrar varios parámetros que se guardan y luego se utilizan para la recuperación de información. Los archivos de audio se almacenan en un servidor de medias, con el uso de un gestor de base de datos se puede realizar consultas que contienen las direcciones físicas de los audios para poder acceder al archivo y efectuar las operaciones deseadas sobre el mismo.

A los ficheros de audio no se le aplica un control de calidad en el momento de entrar al sistema, lo que conlleva a que no cuenten en algunos casos con la calidad necesaria para ser reproducidos. Existe también la posibilidad de que aparezcan duplicados en la base de datos pues solo se verifican sus metadatos básicos, como el nombre, fecha de entrada al sistema, el autor y el tamaño del archivo. Debido a estas condiciones existe desorganización en los archivos, empleo de espacio en disco innecesario y descontrol de la información que se almacena.

Para darle solución a todos estos problemas en la gestión del audio de los sistemas de información audiovisual se elabora un flujo para su procesamiento automático, de manera tal que solucione la falta de control existente.

Resultados obtenidos en el trabajo precedente

Como una primera propuesta de solución a la problemática existente en los sistemas de información audiovisual del departamento Señales Digitales se define un flujo de procesamiento que cuenta con tres fases fundamentales. Primeramente se realizaría una com0probación del formato de audio que va a entrar al sistema. Luego se confeccionaría una huella digital de audio para identificar unívocamente a cada audio en el sistema. Y como último paso sería el reconocimiento donde se le informará al usuario si el audio que se procesa se encuentra o no en el sistema. De encontrarse en el sistema además de informarse al

usuario también se termina el procesamiento, en caso contrario se almacena el audio en la base de datos de media y su huella en la base de datos de huellas digitales y termina el flujo de procesamiento (Cruz, 2011).

Al analizar la propuesta de solución antes abordada se evidencia que esta se encuentra enfocada solamente a la plataforma televisiva PRIMICIA y no a todos los sistemas de información audiovisual donde se manejen archivos de audio, lo que constituye una desventaja de la solución planteada. Además funcionalmente comprende un flujo, muy bien elaborado y justificado, pero que establece una dependencia entre cada acción que se realiza, no se pueden realizar acciones por separado sin tomar en cuenta el orden de ejecución. Ejemplo de esto es la confección de la huella digital, la cual debe haber pasado por una extracción de parámetros de calidad y de haber sido necesario una codificación previa para su confección. Esto trae consigo una falla técnica de ser necesario realizar funciones por separados sin tener que completar el flujo definido.

En cambio el flujo de procesamiento de audio posee una descripción funcional completa que permite un manejo profundo de estos archivos con la obtención de mejoras significativas que propician un manejo adecuado de esta media en los sistemas de información. Las fases que plantean se completan significativamente con el objetivo de lograr que los audios alcancen una calidad óptima además de ser únicos en el sistema. El patrón de calidad propuesto evidencia las características esenciales que deben tener los archivos de audio para que sean agradables antes los oídos humanos en su rango de frecuencia perceptibles. El algoritmo seleccionado para eliminar el duplicados de los archivos presenta un alto respaldo investigativo y una documentación completa que facilita el entendimiento del mismo.

En la presente investigación se propone una solución que elimine las principales inconformidades encontradas en la solución anteriormente analizada. Se pretende realizar un componente de procesamiento digital de audio que garantice un manejo correcto de los archivos sonoros dentro de cualquier sistema de información audiovisual. Para lograr este cometido se requiere un desarrollo independiente de las funciones que contiene el flujo antes descrito, donde una no dependa de la otra y el sistema tenga libertad sobre lo que desee hacer sin seguir un flujo o patrón determinado. El patrón de calidad se establece como se definió desde un inicio pero dándole al sistema la posibilidad de elegir su propio patrón en caso deseado.

El definir una forma diferente y dinámica de implementación se garantiza que la solución presente escalabilidad, flexibilidad, portabilidad, independencia y una reutilización muy provechosa que lo hacen un producto más completo y recomendable.

1.3 Marco teórico sobre componentes de procesamiento digital de audio

En los sistemas de información encargados de la transmisión de audiovisuales hay un problema creciente que cobra fuerza principalmente en las aplicaciones libres, este problema se refiere a los archivos duplicados en las plataformas, que ocupan espacio innecesario en los servidores y sin contar con la calidad y estandarización requerida. La disponibilidad cada vez mayor de sistemas que gestionan el manejo de los audios como archivo multimedia ha creado la necesidad de buscar estrategias y soluciones que pongan fin a este problema tan sustancial para las aplicaciones que manejan archivos de medias.

En el mundo existen soluciones informáticas que tienen como fin la eliminación y gestión de los archivos repetidos en los sistemas informáticos, un ejemplo de ello es **Duplicate Cleaner**, un software privativo que es un sofisticado buscador de archivos duplicados con multitud de opciones para configurar el escaneo y comprobación de duplicidad, pero tiene la desventaja técnica de utilizar metadatos básicos como (CNET, 2009):

- Tamaño
- Nombre del archivo
- Fecha de creación
- Fecha de modificación
- Para los archivos de música: artista, título, álbum

Con la utilización de este software no se soluciona el problema planteado ya que solo verifica metadatos externos de las medias que no expresan si el proceso que se lleva a cabo es correcto, por tal motivo no es posible contar con esta solución para el problema en cuestión. Sin embargo, resulta útil el uso de metadatos simples como un filtro básico de la comprobación de duplicados antes de pasar a un proceso más costoso computacionalmente, siempre y cuando se garantice en el sistema un uso correcto de estos metadatos por los usuarios encargados de suministrar la información.

Otro producto es **NoClone**, también privativo, que se encarga de buscar todos los archivos duplicados que estén en un computador y los eliminará (jpg, gif, videos olvidados, canciones repetidas, etc), con la

posibilidad de restaurar los archivos en caso de error (NoClone, 2007). De igual manera los metadatos utilizados son superficiales y no permiten comprobar las características internas de los archivos.

Otro caso similar es la solución **DoubleKiller** el cual es un programa que hace un completo análisis del disco duro o carpetas seleccionadas del computador, con la ubicación de los ficheros idénticos y la eliminación de los mismos de una manera cómoda y fácil (Geekgt, 2009). **CloneSpy**, de igual manera, es una utilidad para detectar y eliminar archivos duplicados, vacíos o idénticos pero de diferentes versiones, con la recuperación de espacio libre en el disco duro (CloneSpy, 2012). Como este otros productos informáticos se encargan de la misma función sin potenciar las necesidades existentes en los sistemas de procesamiento digital de audio a gran escala.

Con el avance de las tecnologías las huellas digitales de audio han dado un acabado a la búsqueda de la mejor alternativa para eliminar el duplicado de los audios en los sistemas de información audiovisual. Muchas son las investigaciones que se han desarrollado del tema como es el caso del Grupo de Investigación en Tecnología Musical (MTG) de la Universitat Pompeu Fabra que posee el Institut Universitari de l'Audiovisual de Barcelona donde se realiza la experimentación sobre los aspectos relacionados con los medios digitales de comunicación, la cognición y la percepción humana y como esta se puede incorporar a los sistemas digitales (l'Audiovisual, 2009).

La investigación precedente evidencia la existencia de algoritmos como MFCC y VC que proporcionan una solución ventajosa al problema del duplicado de los archivos de audio. Los coeficientes cepstrales de las frecuencias mel son muy utilizados en las investigaciones sobre la representación de la voz o las señales de audio donde se tiene en cuenta la percepción aditiva de los seres humanos (Neira, 2010; Carrasco, 2002). Mientras que el algoritmo de vectores de croma analiza con mayor profundidad el contenido melódico con énfasis en las notas musicales (Goto, 2003).

Al analizar otros problemas presentados por los archivos de audio que interactúan en los diferentes sistemas de información se encuentra la baja calidad con que son transmitidos ya que sus características no son conocidas por el sistema que los almacenan, por lo que no pueden ser codificados a un mismo estándar de calidad. Al ser este un problema evidente han surgido bibliotecas como MediaInfo que permiten obtener informaciones de los archivos de audio que se analizan y FFmpeg que se utiliza para procesar la señal de audio según se desee, brindando variadas formas de hacerlo de manera muy sencilla.

Las soluciones estudiadas no resuelven el objetivo general de esta investigación, por tal motivo es necesario desarrollar un componente de software que elimine los problemas en la gestión de los audios en los sistemas de información audiovisual en el departamento Señales Digitales.

1.4 Metodología de desarrollo a utilizar

Una metodología de desarrollo de software representa un marco de trabajo que tiene entre sus funciones guiar, planificar, estructurar, controlar, manipular y dirigir el proceso de desarrollo de sistemas de información. Surge ante la necesidad de trabajar mediante el uso de procedimientos, técnicas, herramientas y documentos durante el desarrollo de software. Las metodologías se clasifican en dos tipos: ágiles o ligeras y pesadas o tradicionales. Las ágiles tienen como principios el trabajo en equipo como arma fundamental; el avance del trabajo enmarcándose solamente en los elementos necesarios que este exige; la constante interacción con el cliente haciéndolo parte del equipo de trabajo; la posibilidad de cambiar todo lo que debe ser cambiado de forma que se alcance la mayor fiabilidad y calidad en el producto que se desarrolla.

Programación Extrema (*Extreme Programming*, XP)

XP es una metodología de desarrollo de software ágil que potencia las relaciones interpersonales del equipo de trabajo confiándole a esta unión la clave del éxito en el desarrollo. El cliente juega un papel fundamental y decisivo durante el proceso, la retroalimentación entre él y el equipo de trabajo permite determinar de qué forma se va a implementar el trabajo durante todo el proceso de construcción del software. XP se define principalmente para proyectos cambiantes donde el trabajo constante traerá consigo cambios en el desarrollo de la solución para lograr siempre una mejor alternativa (Canós, y otros; Kniberg, 2007).

Proceso Unificado de Desarrollo (por sus siglas en inglés RUP)

Por su parte las metodologías pesadas se centran en el detalle de cada proceso y tarea que se debe desarrollar, en las herramientas a utilizar, genera una documentación extensa que debe justificar a cada paso de avance y además adelanta la puesta en práctica de la solución. Se aplica fundamentalmente a proyectos grandes para realizar en igual período de tiempo y uso de recursos. La metodología pesada que mayor relevancia y utilización posee en el campo del desarrollo de software es RUP. Define fases,

principios, etapas o flujos de trabajo, disciplinas de soporte, entre otros elementos, para mejorar y organizar el trabajo desde el comienzo (Jacobson, y otros, 2000).

Metodología de desarrollo de software seleccionada

Para darle solución a la situación problemática existente en el departamento Señales Digitales se utiliza la metodología de desarrollo de software XP. Entre sus particularidades se define que el cliente debe estar en constante comunicación con el equipo de trabajo. Esto es una ventaja muy amplia porque el cliente participa y forma parte de las decisiones que se tomen en el proceso de desarrollo. El equipo es pequeño y cuenta con una sola persona que es la encargada de todo el desarrollo del producto la cual tiene para su realización un corto período de tiempo. XP permite que estas particularidades no sean un problema, sino un elemento esencial para su aplicación. XP posee funciones que no necesitan generar todos los diagramas existentes sino solamente los que el equipo de desarrollo considere apropiados

RUP está pensado para equipos grandes, documentación extensa, iteraciones amplias, que imposibilitan adaptarse debido a las condiciones existentes. Tiene como elemento base el uso del Lenguaje Unificado de Modelado (por sus siglas en inglés UML) para modelar las funciones del software.. Mucha de la documentación que genera RUP no se utiliza al finalizar el desarrollo del producto, por lo que el tiempo empleado en la realización se pierde. Aunque las diferencias son muy amplias también se tienen puntos en común, ambas metodologías se centran en la arquitectura y son iterativas.

Siendo analizada la situación del departamento Señales Digitales y las características de cada tipo de metodología, se afirma nuevamente el empleo de XP como metodología de desarrollo de software para guiar el trabajo en la solución del problema existente.

1.5 Tecnologías a utilizar en el desarrollo de la solución

En Cuba se hace vital el empleo de GNU\Linux para impulsar el desarrollo informático y el ingreso de bienes que el software propietario no posibilita. Debido a esto se hace necesario utilizar herramientas con tecnología libre para el desarrollo exitoso de esta investigación.

1.5.1 Framework de desarrollo

Un framework o marco de trabajo tiene como fin ser utilizado por otro proyecto de software para ganar en organización y desarrollo, facilita funciones definidas que enriquecen el trabajo. Puede incluir lenguaje interpretado, bibliotecas, soporte a programas y otras herramientas para unir los componentes que sustentan un proyecto. Para GNU\Linux existen frameworks de desarrollo que cumplen la condición de software libre, el más destacado de ellos es Qt.

Qt como framework de desarrollo

Qt es un framework multiplataforma que se emplea en mayor medida para el desarrollo de aplicaciones con interfaz gráfica o de programas como herramientas. Fue construido con el uso de C++ y tiene soporte para lenguajes de programación como: C++; Python por contar con PyQt, PySide y PythonQt; Ruby por contar con QtRuby; PHP por contar con PHP-Qt; Java por contar con QtJambi y D por contar con Dqt. Puede ser utilizado tanto para realizar aplicaciones de escritorio como aplicaciones web, así como para dispositivos de tecnología móvil. Qt tiene entre sus particularidades elementos que sobresalen como son: su framework para aplicaciones multimedia, llamado Phonon; su módulo para aplicaciones 3D con OpenGL; el soporte para la comunicación entre procesos en tiempo de ejecución, entre otras muchas capacidades (Muñoz, 2011).

Con el fin de lograr el desarrollo de una aplicación que cumpla con las necesidades del cliente se utiliza como framework de desarrollo Qt. Una característica muy importante que maneja Qt es el paradigma señales y *slots*, también visto como eventos. Esto es un mecanismo a través del cual los objetos de una aplicación se comunican. Este mecanismo es una de las características distintivas de Qt. Además de brindar facilidades de uso, este framework posee una documentación extensa que posibilita el entendimiento, aprendizaje y desarrollo de su utilización para los usuarios. En la UCI existen grupos de desarrollo que utilizan Qt para todos los procesos que realizan, esto garantiza que existan muchas fuentes de consulta y una documentación extensa sobre Qt, además de aplicaciones que son desarrolladas por estos estudiantes innovadores que evidencia las facilidades que brinda Qt como framework.

Entre las aplicaciones más importantes que utilizan Qt se pueden mencionar: *Google Earth*, *Opera*, *Skype*, *VLC Media Player*, *Virtualbox*, *Psi*, *Doxygen*, *MythTV* y el entorno de escritorio *KDE*. Como último elemento se puede plantear que la portabilidad que posee Qt permite que pueda funcionar en diversas

plataformas ya sean sistemas tipo *Unix* con el servidor gráfico *X Windows System* (Linux, BSDs, Unix), para *Apple Mac OS X*, para sistemas de *Microsoft Windows*, para Linux Embebido e incluso para dispositivos que utilizan *Windows CE*. Dependiendo del entorno donde se haga la compilación, el módulo qmake hará su función y generará los archivos necesarios para que el programa pueda ser ejecutado en el entorno donde se está trabajando sin necesidad de adaptar el código (Ramiro, 2011).

1.5.2 Lenguaje de programación

Cuando se habla de lenguaje de programación se hace referencia a un idioma artificial que está diseñado para la interacción entre un usuario y un computador de forma tal que pueda emplearse para crear programas, algoritmos, que solucionen problemas tales como la comunicación humana. Estos lenguajes pueden utilizarse para crear aplicaciones que cumplan un objetivo dado, desde el menos complicado hasta uno de los más controversiales en la actualidad como es la función de comunicarse como si fuera un ser humano. En su estructura se definen símbolos, reglas sintácticas y semánticas que justifican su comportamiento. Tiene procesos asociados como la depuración, la compilación, la escritura, que unidos en su conjunto forman el concepto de programación.

Para la selección del lenguaje que se utiliza en el desarrollo de esta investigación se definen parámetros a cumplir como: familiarización con el lenguaje, el IDE que lo soporta, tipo de lenguaje, eficiencia y consumo de memoria principalmente. Existen dos tipos de lenguajes de programación, los lenguajes de bajo nivel y los de alto nivel. Los de bajo nivel tienen asociados una sintaxis muy difícil de entender, muy parecida al lenguaje máquina, por lo que es complejo utilizarlos. Sin embargo los lenguajes de alto nivel poseen una sintaxis más cercana al lenguaje humano, por lo que su comprensión de hace más fácil. Entre los lenguajes de alto nivel que tienen mayor popularidad en la programación se encuentran: Java, C#, Python, C, C++, entre otros.

Java como lenguaje de programación

Java es un lenguaje de programación orientado a objeto desarrollado a principio de los años 90. En su composición tiene elementos de C y C++, más simple sin hacer uso de herramientas de bajo nivel. La gestión del uso de memoria está a cargo del propio lenguaje y no del programador. Su desventaja fundamental consiste en que al crearse por la Sun Microsystems esta no permite el acceso al código fuente de Java, solamente unos pocos pudieron mejorarlo, debido a esto su uso es cuestionable.

Actualmente el soporte de Java cae sobre su máquina virtual ya que ofrece portabilidad entre plataformas. Su consumo de memoria es muy alto y las aplicaciones que se realizan con este lenguaje son muy lentas, obliga a los clientes a instalar la máquina virtual en los dispositivos que utilicen una aplicación sobre este lenguaje (Curtis, 2009).

Python como lenguaje de programación

Python es un lenguaje de programación orientado a objetos y de alto nivel, uno de los más populares usado por empresas como Google. Es usado para realizar tanto aplicaciones de escritorio, en su mayoría, como aplicaciones web. Es un lenguaje interpretado y además multiplataforma. Posee licencia de código abierto y es compatible con la licencia pública general de GNU a partir de la versión 2.1.1. Su principal desventaja es su lentitud en tiempo de ejecución, por la misma condición de ser un lenguaje interpretado y no compilado. Es soportado por algunos IDEs de programación como Eclipse y Geany, que comparado con otros lenguajes iguales de relevantes se queda muy por detrás (Knowlton, 2009).

C++ como lenguaje de programación

C++ es un lenguaje de programación que soporta la programación orientada a objetos y la programación estructurada, se le considera un lenguaje híbrido. Fue creado a mediados de los años 80 con la intensión de extender al lenguaje C pero con mecanismos de mejora que permitieran la manipulación de objetos. Actualmente es uno de los lenguajes más potentes y esto se debe en gran medida a que es un lenguaje compilado lo que lo hace eficiente. Otra de sus particularidades es que brinda la posibilidad de redefinir los operadores y de crear nuevos tipos de datos. Para su programación mediante el empleo de herramientas libres existen varios IDEs que lo soportan como es el caso de Eclipse y QtCreator (Blanchete, y otros, 2008).

Lenguaje de programación seleccionado

Al analizar las características fundamentales de cada lenguaje se ha llegado a la conclusión de que el lenguaje de programación idóneo es C++. Se necesita un lenguaje que sea rápido, eficiente, que brinda funciones al equipo de desarrollo de forma tal que la construcción del software sea fluida y sin trabas. La condición de multiplataforma y además multiparadigma marca una diferencia notable entre los demás

lenguajes expuestos. El hecho de ser compilado y no interpretado le da una condición de rapidez y liderazgo sobre los demás.

Para programar en C++ en GNU\Linux el compilador más utilizado es el g++. Este compilador pertenece a la familia de compiladores del proyecto GNU llamada GCC, del inglés *GNU Compiler Collection,* dicha colección está compuesta enteramente por software libre y hoy por hoy se considera un estándar para las plataformas basadas en Unix. Sin la creación del GCC del Proyecto GNU en la actualidad no se pudiera contar con las distribuciones de GNU\Linux que hoy están disponibles ni miles de aplicaciones de software libre. Para programar en C++ sobre GNU\Linux existen varios IDEs de renombre, entre ellos el Eclipse y el QtCreator (Urbino, 2010). Otro elemento a su favor es que la documentación que se encuentra del mismo es amplia debido al desarrollo que ha tenido en la comunidad de usuarios a nivel mundial.

1.5.3 Entorno integrado de desarrollo (por sus siglas en inglés IDE)

El avance de la investigación brinda la posibilidad de escoger un IDE de programación basándose en elementos como el framework de desarrollo y el lenguaje de programación. Para la selección del IDE se establecen parámetros a cumplir: compatibilidad con GNU\Linux, soporte para el lenguaje C++, buen completamiento de código, integración con bibliotecas de Qt, consumo de memoria reducido y por último abundante documentación. Por estos criterios contamos con dos opciones de selección del IDE y estas son: QtCreator y Eclipse.

Eclipse como IDE

Eclipse es un IDE multiplataforma que puede funcionar con varios lenguajes de programación mediante plugins, que añade soporte de lenguajes diferentes. Tiene concebido plugins para C, C++, Ada, Perl, Python, PHP, entre otros. Es de código abierto lo que permite que se mejore en el tiempo. Su principal dificultad se encuentra en el alto consumo de memoria que exige, depende de la máquina virtual de Java lo que es un impedimento y un problema para los usuarios que lo utilizan. Se vale destacar que es uno de los IDEs más utilizado por la comunidad del software libre ya que puede ser muy versátil e interactivo (Eclipse, 2012).

QtCreator como IDE

QtCreator es un IDE multiplataforma creador por Trollthech para el desarrollo de aplicaciones con bibliotecas Qt y distribuido por Nokia como parte del Qt SDK (equipo de desarrollo de software de Qt), que es un grupo de herramientas para desarrollar con el uso del framework Qt. Posee un avanzado editor de código C++, lenguaje que utiliza de forma nativa. Al utilizar el lenguaje de programación C++, se hace uso de la programación orientada a objetos. QtCreator posee herramientas como: *Qt Assistant*, que permite interactuar con la documentación de Qt, donde se muestran todos los elementos que se deben dominar para el trabajo con el IDE; *Qt Designer*, permite crear las interfaces graficas de usuario; *Qt Linguist*, permite crear aplicaciones con soporte para varios idiomas. *Qt Creator* puede generar un paquete de instalación cuando un dispositivo móvil se conecta a un PC, al conectarlo se crea un código y lo ejecuta. (Ramiro, 2011).

Los sistemas operativos que soporta en forma oficial son (Muñoz, 2011):

- GNU/Linux 2.6.x, para versiones de 32 y 64 bits con Qt 4.x instalado. Además hay una versión para Linux con gcc 3.3.
- Mac OS X 10.4 o superior, requiriendo Qt 4.x.
- Windows XP, Vista y 7, requiriendo el compilador MinGW y Qt 4.4.3 para MinGW.

IDE seleccionado

Según el desarrollo de la investigación y los diferentes criterios que se han manejado sobre los entornos de desarrollo Eclipse y QtCreator se puede evidenciar las potencialidades que poseen ambos, pero el equipo de trabajo ha decidido utilizar QtCreator. El uso de memoria es mucho menor con el uso de QtCreator que con Eclipse. QtCreator posee un mayor acabado y una amplia documentación que permite conocer más sobre su funcionamiento y utilización. Su completamiento de código es rápido donde se tiene como referencia la versión 4.7.0. Esta versión ofrece muchas mejoras con respecto a las versiones anteriores de la serie de Qt 4. La característica clave de las versiones Qt 4.7 y *Qt Creator* es *Qt Quick*. Una tecnología de Interfaz de Usuario (UI) de alto nivel que permite a los desarrolladores y diseñadores

de interfaces de usuario trabajar juntos para crear aplicaciones ligeras, animadas e interfaces de usuario táctil (Muñoz, 2011).

1.5.4 Herramienta CASE para el modelado

Las herramientas CASE son programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software (Alfaro, 1999).

Este tipo de herramientas facilitan el proceso de desarrollo de software con el aumento de la productividad y la reducción del tiempo que se utiliza. Utilizándolas se puede diseñar, implementar a partir del diseño realizado, compilar automáticamente, detectar errores y brindarle seguridad al equipo de trabajo sobre el avance de la solución. Para la selección de la herramienta más apropiada se han determinado una serie de aspectos que debe cumplir: modelar el software a través de diagramas UML, generar código en el lenguaje que se decida utilizar y la integración con diferentes IDEs.

Visual Paradigm para UML

Según los requisitos que debe cumplir la herramienta CASE a utilizar en el desarrollo se ha decidido emplear Visual Paradigm para UML. Es una herramienta profesional que soporta el ciclo de vida del desarrollo del software y posibilita un ahorro considerable de tiempo y una calidad óptima en el proceso. Se considera muy completa y fácil de usar, con soporte multiplataforma y excelentes facilidades de interoperabilidad con otras aplicaciones. Posibilita la captura de requisitos, análisis, diseño e implementación para una aplicación en desarrollo.

Entre las ventajas que ofrece están (Sierra, 2006):

- Soporte para UML versión 2.1: con la selección de la metodología XP esta característica es muy provechosa ya que se necesita modelar los diagramas con el uso de UML.
- Generación de código: Modelo a código, diagrama a código, para diferentes lenguajes, entre ellos C++, Java y exportación como HTML.
- Generación de bases de datos: permite la generación automática de bases de dato a partir de un Modelo entidad-relación.

- Interoperabilidad entre diagramas: permite a partir de un diagrama obtener otro que guarde relación con el mismo.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Lo anteriormente expuesto demuestra que Visual Paradigm es la mejor alternativa de herramienta CASE para un desarrollo exitoso del componente que se implementa ya que agilizará el proceso de desarrollo y generará los estereotipos necesarios con la estructura y relaciones deseada.

1.6 Conclusiones parciales

Durante este capítulo se trataron los aspectos significativos de la investigación que serán el soporte teórico de la futura solución y demuestran la necesidad de la realización del presente trabajo. La descripción del procesamiento digital de audio y de la manipulación de medias en los SI audiovisual, demuestran la necesidad de utilizar componentes de software con el fin de solucionar los problemas detectados en los SI del departamento Señales Digitales.

El estudio del estado arte sobre otros sistemas similares evidencia la inexistencia de aplicaciones, componentes, sistemas que solucionen el problema científico expuesto a lo largo de la investigación.

Las condiciones de trabajo sobre la que está respaldada la presente investigación conllevan al uso de XP como metodología de desarrollo. Al tener en cuenta las políticas del departamento Señales Digitales y las particularidades de la solución propuesta se decide utilizar el framework Qt, el IDE Qt Creator y el lenguaje de programación C++, además de Visual Paradigm como herramienta CASE.

CAPÍTULO 2: PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA

El presente capítulo presenta la forma que debe poseer el sistema. Se exponen las principales características y cualidades del sistema a implementar mediante la identificación de los requisitos funcionales y no funcionales con los que debe cumplir la solución final para resolver la situación problémica de la presente investigación. Además se abordan las dos primeras etapas que propone XP: la exploración y la planificación, las cuales marcan el camino que debe seguir el software y el éxito del mismo dependerá de las decisiones que se tomen en estas dos importantes etapas.

2.1 Propuesta de solución

En la presente investigación se desarrolla un componente para el procesamiento de audio en sistemas de información audiovisual que le permita al departamento Señales Digitales del centro de desarrollo GEYSED realizar transformaciones sobre los archivos de audio para comprobar la calidad del mismo y mejorar sus características en caso de este encontrarse en mal estado, también permitirá confeccionar una huella digital de audio como característica unívoca de estos para realizar comprobaciones de duplicado en los sistemas de información.

El componente se integrará a los sistemas de información como un plugin a través de un sistema de plugins que maneje sus funciones como procesos de un mismo sistema. Esto posibilitará que cada una de las funciones que realice el componente pueda ser accedida de manera independiente lo que facilita el trabajo con el mismo sin tener que seguir un flujo predeterminado de acciones sobre los archivos de audio.

Se propone realizar un conjunto de plugins que solucionen las deficiencias presentes en los SI del departamento Señales Digitales. Para alcanzar el objetivo de la presente investigación se confeccionan varios plugins con las siguientes funciones:

- Permitir al sistema conocer cuáles son las características de cualquier audio que desee manipular.
- Determinar si el audio que entra al sistema posee la calidad óptima para que sea parte del mismo, esta calidad puede estar determinada por el sistema o por un patrón de calidad establecido en el proceso de análisis.

- Codificar todo aquel archivo que no le reporte los beneficios normados y cause problemas al ser manipulado; y por último confeccionar una huella digital de audio que posibilite identificar los archivos que se encuentran duplicados en el sistema.

Cada plugin que se desarrolle debe tener como especificidad ofrecer al sistema la información del porciento de realización que se efectúa sobre la media tratada además de la posibilidad de cancelar el proceso que se ejecute por el plugin. Para una mejor integración de los plugin desarrollados con cualquier SI que lo necesite se aplica la programación mediante hilos de procesos en el caso particular de la codificación digital de audio, donde la ejecución de procesos simultáneamente posibilita un rendimiento eficiente del sistema y un tiempo de cómputo menor al que resultaría de una ejecución proceso a proceso.

2.1.1 Extracción de las características de los archivos de audio

Siempre que se desee incluir a un archivo de audio en un sistema de información que posee un contacto directo con usuarios es necesario garantizar que esta media tenga las características acordes para ser reproducida.

Para lograr conocer si un archivo multimedia se corresponde con lo que el sistema de información requiere es necesario extraer sus características más representativas que aporten la información deseada. Esto se hace posible a través de un plugin que ofrece todas las características acústicas que contiene el archivo de audio. Esta ventaja se logra con el uso de la biblioteca MediaInfo que permite la extracción de cada una de las propiedades de los audios. Aunque se aclaró que el sistema puede escoger qué información es la que desea conocer, es válido establecer una serie de datos que facilitarán el conocimiento sobre las condiciones de un audio para ser parte de un sistema de información.

Entre los parámetros que más información aportan sobre un archivo de audio están:

- Número de canales (*Channel*): que se corresponde con el número de señales de audio simultáneas, traducidas en “mono” cuando posee un solo canal donde los sonidos se agrupan en una sola señal y “estéreo” o “multicanal” cuando posee dos canales se consiguen muchos tipos de sonidos.

- Tasa de bits (*BitRate*): hace referencia a la velocidad de transferencia de datos, donde se tiene en cuenta la cantidad de datos que se transmiten en unidad de tiempo por un sistema.

- Frecuencia de muestreo (SamplingRate): se puede describir como la cantidad de muestras por unidad de tiempo de una señal de audio, mientras mayor sea la frecuencia de muestreo mayor calidad tendrá el archivo de audio y su semejanza con el audio original será considerable.

Existen otros parámetros que se pueden extraer de un archivo de audio como el formato, el códec que posee y la profundidad de bits, los cuales también aportan datos concretos de estos archivos.

2.1.2 Comprobación de la calidad de un archivo de audio

Que un archivo de audio posea calidad es una condición sumamente necesaria para su utilización en un sistema de información audiovisual. Resulta muy difícil reproducir un archivo de audio sin calidad, la percepción humana es tan amplia que capta los ruidos y las distorsiones que emite un audio en mal estado.

Cuando se analiza un archivo de audio muchas veces no basta con escuchar el sonido que produce, sino que es necesario adentrarse en las características que lo definen para identificar si se encuentra dentro de los medidores de calidad establecidos para un archivo de audio. (Cruz, 2011) define un patrón de calidad que permite definir si el audio que entra al sistema está apto para ingresar en él, este patrón utiliza la frecuencia de muestreo, el número de canales y la tasa de bits.

Al analizar la frecuencia de muestreo se plantea que su valor estándar es de 44.1 KHz, por lo que para que un archivo tengo una calidad óptima debe tener un valor aproximado a este. La tasa de bits se puede identificar de dos formas: variable o constante, la diferencia de una con otra está dada por la representación de la densidad de información acústica, cuando se utilizan archivos de audio normalmente se elige una tasa de bits variable para conseguir mayor calidad al poder aumentar o disminuir la tasa de bits dinámicamente en dependencia de la complejidad de la música en cada momento. El número de canales puede ser mono o estéreo, la diferencia entre uno y otro está dada por las sensaciones acústicas que se logran, por tal motivo en la mayoría de los casos se trabaja con audio estéreo.

Con el fin de detectar los archivos de audio que no poseen calidad se confecciona un plugin que verifique las características que posee un audio al entrar al sistema. Estas características se comparan con los valores acústicos que el usuario del sistema definió para que cada audio se encuentre estandarizado a un mismo patrón de calidad. Si en un futuro se demuestra que la calidad óptima para la frecuencia de muestreo ya no es 44.1 KHz como plantea (Cruz, 2011), el sistema tiene la posibilidad de comparar con

un patrón que el usuario establezca a través de él. En la realización de este plugin se ve reflejada la utilización del primer plugin realizado Extraer Características para comprobar cuáles son las capacidades que contiene el audio a analizar. De esta forma se hace efectiva la ejecución del plugin y consume menos tiempo de cómputo al no tener que volver a implementar las funciones de extracción de características sino con un plugin implementado que posee esa función.

2.1.3 Codificación de un archivo de audio

La codificación de un archivo de audio es un proceso necesario en gran medida para poder obtener como producto final una media con mejores estadísticas de calidad en la reproducción y utilización. Un archivo que es sometido a este proceso es porque alguno de sus parámetros no se encuentra con los valores requeridos por el sistema y es necesario procesarlo para lograr los valores deseados.

Dándole solución a este problema tan común en muchos archivos de audios se implementa un plugin de codificación de audio que posibilite modificar los parámetros que el sistema notifique. Para lograr este resultado se hizo uso de la biblioteca ffmpeg la cual proporciona una serie de funciones que permiten modificar la característica seleccionada por el sistema de información que manipula tanto un archivo de audio como de video, pero en esta investigación solo se abordará el tema del audio. La biblioteca FFmpeg está desarrollada bajo Linux, pero puede ser usada en sistemas operativos como Windows.

A continuación se muestra cómo se debe utilizar la biblioteca ffmpeg en el procesamiento de un archivo de audio:

- ffmpeg este comando es para declarar el comienzo de la codificación
- –i representa la entrada de un archivo para ser codificado

Seguido de estos comandos comienzan a ponerse las características que se desea modificar con los valores propuestos:

- –ar modifica la frecuencia de muestreo, sino se especifica valor él establece por defecto 44100 Hz;
- –ab modifica la tasa de bits, sino se especifica valor se establece por defecto en 64 k.

- –aq establece la calidad del audio en valores de tipo Variable como define una clasificación de la tasa de bit, a mayor valor de aq mayor calidad de audio Variable.

- –ac modifica la cantidad de canales, si no se especifica valor se establece por defecto un canal y el flujo de salida establece por defecto en el mismo número de canales de audio en la entrada.

- –acodec especifica el códec que tendrá el audio analizado, en caso de mantener el que trae por defecto se utiliza el comando copy.

Son muchos los comandos que existen con los que se puede codificar, para cambiarle un parámetro cualquiera solo hay que agregar a la línea de comandos el dato que se desea modificar con el comando que tiene establecido.

2.1.4 Confección de la huella digital de audio

Cuando el archivo de audio se encuentra con la calidad requerida está en condiciones de que se le extraiga la huella digital que lo identificará a partir de ese momento en el sistema. Se hace necesario obtener un conjunto de valores que representen las principales características de las muestras del mismo. Por ello, se considera que la señal es estacionaria en unos pocos de milisegundos. La misma, será dividida en bloques de aproximadamente 25 ms lo que equivale a 1102 muestras para la frecuencia de muestreo establecida (Cruz, 2011).

Con el objetivo de mantener la continuidad de la información de la señal, es muy común que los bloques estén solapados entre sí, para que no se pierdan los eventos que se encuentran en la transición (Saráchaga, y otros, 2006). Debido a esto, el solapamiento será de 15 ms que es equivalente a 661 muestras para la frecuencia de muestreo establecida (Ver Figura 5).

Figura 3: Esquema de sub-división de la señal de audio.

Además de seleccionar la longitud, se debe elegir también el tipo de ventana a aplicar en los extremos de los bloques para evitar discontinuidades de la señal (Cano, 2007). Cada ventana se caracteriza por la forma de sus lóbulos central y laterales en frecuencia. Se requiere de una ventana, que su lóbulo central sea lo más angosto posible y que los lóbulos laterales sean pequeños para tener una buena resolución en frecuencia (Saráchaga, y otros, 2005).

Las ventanas son funciones matemáticas que intervienen en el procesamiento de señales con el fin de evitar la discontinuidad al principio y fin de los bloques en los que se puede dividir una señal de audio. Se utiliza una ventana cuando se necesita un tamaño limitado de la señal para efectuar cálculos sobre ella, si no existe un número finito de puntos no es posible efectuar ningún cálculo matemático sobre ella.

La utilización de una ventana cambia el espectro en frecuencia de la señal. En la actualidad existen sistemas que obtienen el espectro de una señal a través de algoritmos que son implementados en equipos de procesamiento de señales. Un equipo capaz de realizar esta función con algunas limitaciones (tales como la capacidad de procesamiento, el número de muestras disponibles por segundo) es un ordenador (Alvarado, 2005).

Las ventanas que más se utilizan son la Rectangular, la de Hanning, la de Hamming, la de Bartlett y la de Blackman (Saráchaga, y otros, 2006; Cáceres, 2007; Friedrich, 2002).

Por las características de la aplicación se decidió utilizar la ventana de Hamming, la cual no es más que una función coseno que busca acercar a cero o cerca de cero los extremos de los bloques y así reducir el salto (Cáceres, 2007;Friedrich, 2002). Por tanto, su objetivo será evitar que las características de los extremos varíen la interpretación de lo que ocurre en la parte más significativa de las muestras seleccionadas.

Se hace necesario aplicar también transformaciones lineales, es decir, la proyección del conjunto de medidas para un nuevo conjunto de características, de forma tal, que si se elige la transformación correctamente la redundancia se reduce de manera significativa. Para lograr dicho objetivo y para que la señal de audio se haga robusta frente a varios tipos de ruidos y distorsiones, se aplicará el análisis de Fourier (Cano, 2007).

Posteriormente se realiza el proceso de extracción de características, con el uso del algoritmo MFCC. El resultado de este proceso es una secuencia de vectores de parámetros, que cumplen dos objetivos simultáneamente: representar las características esenciales de la señal de voz, que permite su

reconocimiento, y reducir la cantidad de información, lo cual es importante para obtener un buen desempeño en tiempo de cómputo (Maldonado, 2003).

En el trabajo consultado (Neira, 2010) utiliza doce coeficientes en los vectores de características que obtienen. Según un estudio realizado en (Saráchaga, y otros, 2006), si se utiliza un número mayor de doce coeficientes la aproximación es demasiado exacta y con un número menor es demasiado vaga, pero con el uso de doce se aprecia de forma clara. Por tanto, luego de analizar cada uno de los aspectos anteriores se propone que los vectores de características proporcionados por el algoritmo MFCC estén formados por doce coeficientes. Por último, se procederá a realizar el modelado de la huella digital de audio, en el cual se fusionarán dos técnicas de este tipo para asegurar una mayor fortaleza en el proceso de reconocimiento.

La primera técnica es la normalización de la media cepstral que consiste en calcular el valor medio de cada coeficiente cepstral en el audio seleccionado y normalizarlo con la sustracción de ese valor medio. Permite modificar las estadísticas de las características de voz ruidosa y hacerlas coincidir con las de una referencia limpia (Martínez, 2007).

La segunda técnica utilizada es la varianza o media cuadrática de las puntuaciones diferenciales. Para hallarla, se resta a cada valor la media y se eleva al cuadrado; se repite para todos los valores, se suman todos los cuadrados y se divide por el número de valores (Rosales, 2010).

2.1.5 Reconocimiento

A partir de la huella digital de audio obtenida en la fase anterior, debe compararse con cada una de las huellas digitales existentes en la base de datos de huellas digitales. Este proceso requiere de un gran tiempo de cómputo en dependencia de la cantidad de archivos almacenados y se puede realizar de dos formas: en tiempo real y en tiempo no real. En el reconocimiento en tiempo no real, una vez que el usuario accede al mismo, no tendrá una respuesta inmediata sobre si el audio que desea procesar se encuentra duplicado. El sistema procesará la petición posteriormente y emitirá una notificación una vez concluido el proceso. En este caso el correcto funcionamiento del sistema no depende del tiempo en que se produce el resultado, solo tiene en cuenta la lógica que posee. Este tipo de reconocimiento puede constituir una limitación en muchas ocasiones cuando se requiera de una respuesta inmediata (Cruz, 2011).

En el reconocimiento en tiempo real tanto el usuario como el sistema aunque estén físicamente separados, coinciden en tiempo, lo que facilita la interacción inmediata. Para este tipo de reconocimiento, a diferencia del que no se hace en tiempo real, además de tenerse en cuenta la lógica que posee el sistema, es sumamente importante el tiempo en que se produce el resultado, donde puede convertirse en crítico (Cruz, 2011).

Después de haber analizado cada una de las características anteriores y con el análisis de las ventajas y desventajas que brindan a los sistemas de información audiovisual ambos reconocimientos, se propone que el sistema brinde la posibilidad de elegir el tipo de reconocimiento a utilizar cada vez que se acceda a esta funcionalidad para que el usuario decida de acuerdo al tiempo que tenga disponible.

Sin importar la elección del sistema, el reconocimiento se podría implementar de manera secuencial, en el cual las comparaciones de las huellas digitales se realizarían una tras otra, de manera consecutiva. Este proceso se agiliza considerablemente al utilizarse la programación paralela, con la cual, se puede compartir la memoria a través de los hilos de reconocimiento y de esta manera ejecutar más instrucciones en menos tiempo, aunque las instrucciones demoren lo mismo en ejecutarse. Además, este tipo de programación, brinda un mayor rendimiento ya que permite sacarle provecho a arquitecturas con múltiples núcleos y le da un mejor aprovechamiento a los recursos con una la latencia menor, incluso en sistemas con un solo procesador (Santos, y otros, 2009). Por el análisis realizado anteriormente, se propone que el reconocimiento en los sistemas se realice con el uso de la programación paralela (Cruz, 2011).

2.2 Identificación de los requisitos del componente de audio

Cuando se comienza a desarrollar cualquier sistema informático es necesario tener en cuenta qué es lo que el usuario final realmente quiere y necesita. Es ineficiente empezar a implementar sin antes tener una definición detallada de los requisitos y capacidades que debe cumplir el sistema que se construye. Los requisitos expresan qué debe hacer el sistema, para su identificación se deben llevar a cabo numerosas entrevistas a los clientes para identificar sus necesidades y sus expectativas con el software, sus ideas sobre el mismo y lo que esperan del trabajo.

Entre las prácticas más usadas para la selección de las principales capacidades y cualidades del sistema están:

- Entrevistas con los clientes: grupo de preguntas con la intención de recabar información sobre un tema. Se entrevistan a personas individuales o en grupos.
- Tormenta de ideas con el cliente: técnica que ayuda a juntar ideas cuando no se está muy seguro de qué es lo más importante a tratar sobre el tema de interés, se recogen muchas ideas inicialmente en un papel y al final se releen con el fin de ordenarlas, la tormenta de ideas debe ser corta de cerca de diez minutos.
- Cuestionarios: son una buena alternativa a las entrevistas, pues pueden ser la única manera de relacionarse con gran cantidad de personas implicadas con el futuro sistema.
- Observación: no hay una forma mejor de saber cómo funcionan los procesos a automatizar que estar dentro del ambiente donde se desplegará el futuro sistema, mediante la observación se exploran mejor los procesos, se revisa que los procesos identificados se llevan a cabo y permite un mejor entendimiento del entorno donde se empleará el software.

En la presente investigación se utilizaron para definir los requisitos del sistema las entrevistas a los líderes de los proyectos que integran el departamento Sistemas Digitales para identificar cuáles eran las principales necesidades que poseían los sistemas de información audiovisual que se producen en cada uno de los proyectos del departamento. Como resultado de estas entrevistas se ha podido obtener una lista con los requisitos que el sistema debe cumplir.

2.2.1 Requisitos funcionales del componente de audio

Los requisitos funcionales ilustran cuál será el comportamiento interno del sistema que se desarrolla. Son las capacidades y condiciones que debe tener el sistema. A continuación se muestran los definidos en esta investigación:

R 1: Extraer características de los archivos de audio

Descripción: El sistema debe permitir extraer los atributos característicos de un archivo de audio del cual se conoce solamente su dirección origen.

Entradas:

- Dirección origen del archivo de audio sobre el cual se va a interactuar para obtener toda la información deseada por el sistema.

- Cadena de parámetros que indiquen qué atributos del audio hay que extraer.

Salidas:

- Cadena contenedora de la información del audio según los parámetros indicados por el sistema.

R 2: Comprobar calidad del audio

Descripción: El sistema debe permitir comprobar la calidad de un archivo de audio del cual se conoce su dirección origen y sus características fundamentales. La forma de realizar esta comprobación es a través de la definición de parámetros que tengan valores apropiados para que un audio pueda ser reproducido y manipulado dentro de un SI y comparar que los parámetros que posee el audio tengan la misma calidad que los definidos.

Entradas:

- Dirección origen del archivo de audio sobre el cual se va a interactuar para obtener toda la información deseada por el sistema.
- Cadena de parámetros que indiquen los atributos del audio a los que hay que comprobarle los valores.
- Cadena de parámetros que especifique los valores que debe poseer cada parámetro a analizar en el audio procesado.

Salidas:

- Verdadero si el audio posee la calidad deseada por el sistema y Falso en caso contrario.

R 3: Codificar archivo de audio

Descripción: El sistema debe permitir codificar un archivo de audio del cual se conoce su dirección origen. Este procesamiento digital del audio consiste en convertir un archivo de audio dándole a cada parámetro del mismo los valores que el sistema desee que tenga para sustituirlos por los contenidos por el audio.

Entradas:

- Dirección origen del archivo de audio para ser procesado por el sistema.
- Cadena de parámetros que indiquen los atributos del audio a los que hay que codificar.

- Cadena de parámetros que especifique los valores que debe poseer cada parámetro que se codifique.

Salidas:

- Archivo de audio codificado según las especificidades hechas por el sistema.

R 4: Confeccionar huella digital de audio

Descripción: El sistema debe permitir confeccionar una huella digital de un audio del cual solo se conoce su dirección origen. La huella no es más que un vector de 12 coeficientes que la identificará unívocamente del resto de los archivos de audio.

Entradas:

- Dirección origen del archivo de audio al cual se le va a confeccionar la huella digital.

Salidas:

- Huella digital de audio.

R 5: Verificar existencia de duplicados

Descripción: El sistema debe permitir verificar si un archivo del audio que se desea incluir en el SI ya se encuentra en el mismo. Para este proceso se hace uso de la huella digital perteneciente al audio analizado, la cual se compara con todas las huellas digitales existentes en la base de datos de huellas para descartar que se encuentre almacenada. En caso de no encontrar una huella semejante a la del audio en cuestión, se almacena el archivo de audio en la base de datos de medias y la huella digital que le corresponde en la base de datos de huellas digitales.

Entradas:

- Dirección origen del archivo de audio que se desea almacenar en el sistema.
- Huella digital del audio analizado.

Salidas:

- Verdadero en caso de que exista una huella digital semejante a la del archivo de audio que se desea incluir en el sistema y falso en caso contrario.

R 6: Estado de procesos

Descripción: El sistema debe permitir conocer el estado de los procesos que ejecutan los plugin desarrollados.

Entradas:

- Señal que solicita conocer el estado en el que se encuentra el proceso que se ejecuta.

Salidas:

- Porciento de ejecución del proceso.

R 7: Cancelar procesos

Descripción: El sistema debe permitir cancelar los procesos que ejecutan los plugin desarrollados.

Entradas:

- Señal que solicita cancelar el proceso que se ejecuta.

2.2.2 Requisitos no funcionales del componente de audio

Los requisitos no funcionales son las características o cualidades que debe tener el sistema que se desarrolla. Son precisiones de los servicios que ofrece el sistema. Existen muchas formas de clasificar a estos requisitos por categorías. A continuación se muestran los definidos en esta investigación:

Software:

- Se requieren las bibliotecas de Qt libqtcore4 y libqt4-sql a partir de la versión 4.7.0.
- Se requieren las bibliotecas MediaInfo 0.7.54 y FFmpeg 0.10.

Restricciones de diseño y de implementación:

- Para el desarrollo se emplearán sólo herramientas libres, como el IDE Qt Creator versión 2.2.1 y las bibliotecas especificadas de Qt versión 4.7.3.
- El diseño del componente debe permitir una fácil adaptabilidad y extensibilidad por si en el futuro se desean incluir nuevas funcionalidades y mejoras. Esto se logra a través de un sistema de plugin independientes unos de otros que permiten incluir o eliminar nuevas funciones sin afectar todo el

sistema. Estas modificaciones son posibles por la utilización de Microkernel como patrón arquitectónico.

Requisitos de soporte:

- Se debe brindar un manual de usuario para facilitar el trabajo a los futuros usuarios de sistema que lo utilice.

2.3 Exploración y planificación

La metodología ágil XP tiene como dos primeras etapas la exploración y la planificación. En la exploración como etapa inicial se describen las historias de usuario y se les otorga una prioridad a las mismas, mientras que en la planificación es donde se elabora el plan de iteraciones y el plan de entregas por los cuales se regirá el equipo de desarrollo. Estas dos etapas son vitales para el desenlace del software debido a que las principales decisiones se toman en ellas, y conducen al éxito o fracaso del producto final.

2.3.1 Exploración

En esta primera etapa de la metodología XP el objetivo fundamental es crear las bases para comenzar a construir el software donde el cliente y los desarrolladores tengan definido lo que el sistema debe hacer. Para lograr un resultado de calidad se toman los requisitos funcionales que se definieron y con ellos se describen las historias de usuario.

2.3.2 Historias de Usuarios

La terminología historias de usuario se asemeja a los casos de uso en la metodología de desarrollo de software RUP, la diferencia consiste en que las primeras tienen una descripción breve que describe lo que debe hacer. Una historia de usuario puede estar compuesta por uno o varios requisitos identificados que escribe el propio cliente como protagonista fundamental del entorno de desarrollo. A cada historia de usuario se le asigna en su concepción una prioridad y duración que determina el equipo de desarrollo. Las historias de usuario realizadas se referencian en el *Anexo # 2* de la investigación. A continuación se presenta un ejemplo de estas:

Historia de usuario "Extraer características de los archivos de audio"

Historia de Usuario	
Número: 1	**Nombre:** Extraer características de los archivos de audio.
Usuario: Sistema	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Estimación: 1	**Iteración Asignada:** 1
Descripción: Permite extraer todas las características que el sistema desee conocer del audio involucrado. Para ello el usuario del sistema solicita la ejecución del plugin especificando el audio sobre el que se va a efectuar la extracción y las características que se desean obtener, las cuales se conocen con el uso de la biblioteca MediaInfo.	
Observaciones: Al extraer las características del audio sistema tiene la posibilidad de poder utilizarlas en los demás procesos que realiza el sistema.	

Tabla 1: HU Extraer características de los archivos de audio.

2.3.2 Planificación

Esta etapa tan importante de la metodología XP trae consigo la toma de decisiones en cuanto a la priorización de los requisitos funcionales, las iteraciones que tendrán, la duración y el esfuerzo, para de esta manera obtener un Plan de Entregas y un Plan de Iteraciones que haga posible el desarrollo en tiempo del sistema.

2.3.3 Iteraciones

Cuando se han definido las historias de usuario y a las mismas se le ha asignado una prioridad en dependencia de la importancia en el futuro sistema es necesario establecer las iteraciones con las que se contarán. Cada iteración consiste en una pequeña implementación donde se construyen las historias de usuario. La ubicación de cada HU en las iteraciones depende de la necesidad que tiene el sistema de realizar una acción antes que otra, en la primera iteración siempre deben encontrarse las HU que representen una prioridad inmediata de realización para el equipo de desarrollo. Se recomienda que todas las iteraciones tengan la misma duración pues establece un tiempo de desarrollo fijo e invariable. Cada

iteración, según XP, debe ser corta, entre 2 y 3 semanas, para la presente investigación se ha decidido utilizar iteraciones de 21 días laborables.

2.3.4 Plan de iteraciones

En el Plan de Iteraciones se colocaron las historias de usuario en dependencia de la complejidad y prioridad definida para las mismas. Su organización se realiza para garantizar que cada una de ellas se relacione con la implementación de las demás dentro de una misma iteración. A continuación se muestran las historias de usuario organizadas por iteraciones:

Iteraciones	Orden de las Historias de Usuario a implementar	Prioridad	Días	Total de días de la iteración
Iteración 1	**1.** Extraer parámetros de calidad	**Alta**	**7**	**21 días**
	2. Comprobar calidad del audio	**Alta**	**4**	
	3. Codificar archivo de audio	**Alta**	**6**	
	4. Confeccionar huella digital	**Alta**	**4**	
Iteración 2	**5.** Confeccionar huella digital	**Alta**	**10**	**21 días**
	6. Verificar existencia de duplicados	**Alta**	**5**	
	7. Estado de procesos.	**Alta**	**3**	
	8. Cancelar procesos.	**Alta**	**3**	

Tabla 2: Plan de iteraciones.

2.3.5 Plan de entregas

El Plan de Entregas es una estrategia que vincula a los clientes con el equipo de desarrollo del sistema, cada cual juega su rol en todo momento. El cliente es el encargado de guiar el plan mientras que los desarrolladores son los que trabajan sobre él. El cliente es el que decide que HU se deben entregar y los desarrolladores establecen el tiempo que pasará para la entrega. Además de lo anteriormente explicado, el plan es el documento mediante el cual los clientes le exigen a los desarrolladores las HU según el tiempo establecido para las mismas. El cumplimiento estricto de este plan es lo que garantiza la moral y el prestigio del equipo de desarrollo frente al cliente.

A continuación se muestra el Plan de Entregas:

Entregable	Fin de la primera iteración (9 de abril)	Fin de la segunda iteración (7 de mayo)
Componente de audio	Versión 0.5	Versión 1.0

Tabla 3: Plan de Entregas.

2.4 Diseño de la arquitectura del componente de audio

Durante el proceso de desarrollo de software se hace vital definir la arquitectura con la que va a contar el sistema que se construye. Para decidir la arquitectura de una aplicación es fundamental hacer antes una estimación de la infraestructura con la que se va a desplegar la solución final. Luego se deben seleccionar y combinar los estilos y patrones arquitectónicos que se emplearán.

La arquitectura de software provee al cliente de una vista al sistema que incluye los principales componentes que lo integran, está afectada por la estructura, el comportamiento, el uso, la funcionalidad, el rendimiento, la flexibilidad, la reutilización, factibilidad de compresión, las restricciones y compromisos económicos y tecnológicos, y la estética (Jacobson, y otros, 1999).

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular, el cual es recurrente dentro de un cierto contexto. Este esquema se especifica con la descripción de los componentes, con sus responsabilidades y relaciones (Ochoa, 2005).

Patrón arquitectónico Microkernel para sistemas adaptables

En la presente investigación se seleccionan los patrones para sistemas adaptables como arquitectura de software a utilizar. Como ejemplo de este tipo de patrón existe el patrón *Microkernel* que es usado en sistemas de software que deben adaptarse a los cambios en los requisitos con la separación del núcleo funcional, la funcionalidad extendida y los aspectos relativos al cliente. Otro ejemplo es el patrón *Reflexion*, el cual provee un mecanismo para cambiar la estructura y el comportamiento de un sistema de software, en forma dinámica donde divide a una aplicación en dos partes: un meta nivel que provee información acerca de las propiedades del subsistema seleccionado, y un nivel base que provee la lógica de la aplicación.

El trabajar sobre el patrón *Microkernel* posibilita que el sistema pueda modificar sus requisitos aunque pase mucho tiempo, esta propiedad permite que el cliente pueda establecer nuevas condiciones que podrán ser cumplidas sin tener que comenzar de cero. Este patrón separa al núcleo de los demás

servicios que presta el sistema; el núcleo es muy pequeño y consume pocos recursos del sistema. En su definición se encuentran 5 tipos de componentes: servidores internos (a los que se accede directamente), servidores externos (establecen la comunicación con los clientes), clientes (se comunica con los servidores externos para realizar peticiones), adaptadores (permiten a los clientes acceder a los servicios de su servidor externo), *Microkernel* (implementa los servicios centrales como facilidades de comunicación o manejo de recursos) (Ver Figura 6).

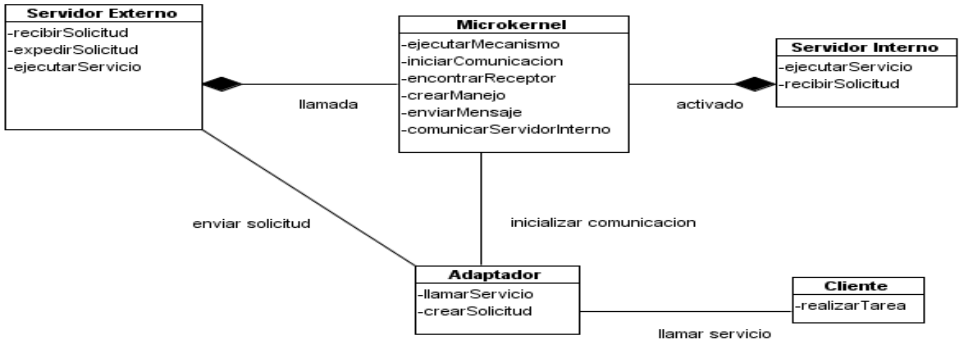

Figura 4: Estructura del patrón Microkernel.

(Buschmann, y otros, 1996)

El usar el patrón *Microkernel* ofrece portabilidad, flexibilidad y extensibilidad al sistema lo que constituye una ventaja muy provechosa para los desarrolladores. También se puede separar en el *Microkernel* las funcionalidades más complejas de las básicas. Permite que el sistema sea escalable ya que puede funcionar de forma distribuida o local y la comunicación entre los diferentes procesos que lo componen lo hace de manera transparente. Aunque las ventajas de su utilización son varias también trae consigo problemas a la hora de definir cuáles serán las funcionalidades básicas y complejas, cuáles son las principales tareas a desarrollar por el *Microkernel* y su diseño e implementación se tornan demasiados complejos en muchas ocasiones.

Para la presente investigación se utilizará el patrón *Microkernel* para representar estructuralmente los plugins que se desarrollan, además del patrón *Reflexión* para lograr la comunicación con el núcleo del sistema y la posibilidad de cambios en el mismo. Además se hará uso del patrón de diseño Observador (en inglés *Observer*) para posibilitar el uso de los plugins por los sistemas de información audiovisuales.

El patrón de diseño que permite distribuir el código realizado en varias direcciones sin tener que modificarlo se llama Observador el cual establece una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos. Su intención fundamental consiste en proporcionar a los componentes una forma flexible de enviar mensajes de difusión a los receptores interesados.

Observer es un patrón de comportamiento que es una de las clasificaciones que pueden tener los patrones de diseño. Está relacionado con algoritmos de funcionamiento y asignación de responsabilidades a clases y objetos. Los patrones de comportamiento describen no solamente estructuras de relación entre objetos o clases sino también esquemas de comunicación entre ellos.

Para emplear el patrón de diseño *Observer* es necesario destacar los elementos que interactúan en el sistema los cuales son: *Subject*, *Observer, ConcreteSubject, ConcreteObserver*.

El objeto observado notifica a sus observadores cada vez que ocurre un cambio. Después de ser informado de un cambio en el objeto observado, cada observador concreto puede pedirle la información que necesita para reconciliar su estado con el de aquél.

2.5 Estándar de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. También se puede entender como un conjunto de reglas de notación y nomenclatura, específicas de cada lenguaje de programación, que se usan y se siguen durante la fase de implementación (codificación) de una aplicación y reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores que no son detectados por lo compiladores, y reducen el tiempo y coste de las actividades de depuración y pruebas necesarias para la detección y corrección de los mismos (Matos, 2008).

La elaboración y el empleo de estándares de codificación ofrecen numerosas ventajas a la hora de elaborar cualquier tipo de producto software y entre sus principales ventajas se pueden mencionar:

- Se reduce la posibilidad de cometer errores.
- Se obtiene un código legible y comprensible.

- Mejora la comunicación entre los miembros del grupo de programadores del equipo de desarrollo.
- Se asegura la legibilidad del código entre distintos programadores, y facilita la compilación del mismo.
- Provee una guía para el encargado de mantenimiento/actualización del sistema con código claro y bien documentado.
- Facilita la portabilidad entre plataformas y aplicaciones.

Para la elaboración de un estándar de codificación se debe tener en cuenta, entre otros aspectos, las especificaciones y características del framework de desarrollo junto al o los lenguajes de programación escogidos. En la presente investigación se propone el uso del siguiente estándar de codificación para seguir con las políticas establecidas por el departamento Señales Digitales:

Comentarios, separadores, líneas y espacios en blancos		

Ubicación de comentarios.	Inicio de una clase y al final de una línea o bloque de código.	Describir brevemente el objetivo (qué función realiza) de una clase previamente de su declaración se desea especificar la acción específica de una línea de código, utilizar un breve comentario al final de esa línea. Ejemplo:
		//Esto es un comentario al inicio de una clase
		class classMediaInfo:public QObject
		{…}
		QString channels; //este atributo indica los canales que tiene el archivo de audio

Líneas en blanco.	Antes y después de cada función.	Dejar una línea en blanco entre las declaraciones de cada función, se puede utilizar entre líneas cuando el bloque de código es extenso para dar claridad a su entendimiento. Ejemplo: void funcionUno()::otraFuncion() {…} //espacio en blanco void funcionDos()::otraFuncion()

		{…}
Espacios en blanco.	Entre operadores aritméticos y lógicos.	Usar un espacio en blanco entre los miembros de operaciones aritméticas y lógicas. Ejemplo: media_i = new Class_MediaInfo(); samplingRate = media_i->GetSamplingRate();
Clases, objetos, funciones, atributos		
Apariencia de clases y funciones.	Se tiene en cuenta las especificaciones del framework.	Se establece usar la notación *Camel Casing* (es una norma de notación que establece que las palabras compuestas debe ser escrita con mayúscula excepto la primera palabra) para nombrar las clases y las funciones. Ejemplo: void comprobarCalidad::obtenerDatosAudio() { ... }
Nombre de clases y objetos.	Relacionados con el propósito de su función.	Nombrar las clases de manera tal que con solo leerla se note su objetivo. Ejemplo: class classMediaInfo:public QObject {…}
Apariencia de atributos.	Letras minúsculas.	Los atributos deben ser escritos en minúsculas y su nombre debe guardar relación con el valor que almacena. Ejemplo: QString channels;

Tabla 4: Estándares de codificación.

2.6 Conclusiones parciales

En el presente capítulo se determinaron las capacidades y características que tiene el sistema desarrollado. Estas capacidades se representaron y agruparon en historias de usuarios, que son la base para el desarrollo de la solución. El cumplimiento de las dos primeras fases de la metodología XP permite la obtención de los artefactos necesarios para el desarrollo de la investigación. La elaboración del plan de entregas por el cual el equipo de desarrollo se debe regir para cumplir con las entregas en tiempo posiciona el desarrollo de las funcionalidades en dos iteraciones fundamentales que completarán el ciclo de implementación.

El análisis de las características del problema a resolver se determina la utilización de una arquitectura para sistemas adaptables, con el empleo de los patrones arquitectónicos: *Microkernel* y *Reflexión*, y como

patrón de diseño *Observer*. La extracción de las características acústicas se realiza a través de la biblioteca MediaInfo. La codificación de los archivos de audio se realiza a través de la biblioteca ffmpeg. La necesidad de realización de la huella digital de conlleva a la utilización de ventanas de Hamming como primer paso y el algoritmo Coeficientes Cepstrales de Frecuencia Mel con una serie de funciones que completan el funcionamiento correcto de la huella obtenida.

Las características del departamento Señales Digitales evidencian la utilización de un estándar de codificación con la notación *Camel Casing* para el desarrollo de las funcionalidades que requiere el sistema.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

En la realización del presente capítulo se abordan dos fases fundamentales en el ciclo de desarrollo propuesto por la metodología de desarrollo de software XP, las fases de implementación y prueba. Se muestran los principales artefactos que genera y la calidad de lo realizado se muestra mediante las pruebas realizadas como última etapa del desarrollo de la solución.

3.1 Bibliotecas utilizadas

3.1.1 Bibliotecas de Qt

Qt brinda facilidades a los programadores que le confieren una alta calidad. Las bibliotecas de Qt ofrecen clases y propiedades que se pueden utilizar para la implementación de diferentes programas. En esta investigación se utilizaron varias bibliotecas, algunas de estas son (Nokia Corporation, 2011):

- **QtSql:** brinda acceso a las clases para el trabajo con base de datos.
- **QString:** proporciona una cadena de caracteres Unicode. QString almacena una serie de QChars de 16 bits, donde a cada QChar le corresponde un Unicode de 4.0 caracteres. Unicode es un estándar internacional que soporta la mayoría de los sistemas de escritura en uso hoy en día. Es un super-conjunto de US-ASCII (ANSI X3.4-1986) y Latin-1 (ISO 8859-1), y todos los personajes US-ASCII/Latin-1 están disponibles en las mismas posiciones del código.
- **QStringList:** La clase QStringList ofrece una lista de cadenas. QStringList hereda de <QString> QList. Al igual que QList, QStringList se comparte de forma implícita. Se proporciona un servicio rápido basado en índices de acceso, así como inserciones y eliminaciones rápidas. Pasa listas de cadenas como parámetros es vez rápido y seguro. Todas las funcionalidades de QList también se aplican a QStringList.
- **QtPlugin:** es una biblioteca que permite establecer macros para definir plugins. Entre los principales macros que define están Q_DECLARE_INTERFACE, Q_EXPORT_PLUGIN2, Q_IMPORT_PLUGIN.
- **QPluginLoader:** carga un plugin en tiempo de ejecución, proporciona acceso a un plugin de Qt. Un plugin de Qt se almacena en una biblioteca compartida (DLL) y comprueba que un plugin está enlazado con la misma versión de Qt como la aplicación.

- **QMainWindow:** proporciona una ventana de la aplicación principal que brinda un marco para la creación de interfaz de usuario de una aplicación. Qt tiene QMainWindow y sus clases relacionadas con la gestión de la ventana principal.
- **QFileDialog:** proporciona un cuadro de diálogo que permiten a los usuarios seleccionar los archivos o directorios, además permite al usuario recorrer el sistema de archivos con el fin de seleccionar los archivos de una o varias o un directorio.
- **QFile:** proporciona una interfaz para leer y escribir a los archivos, es un dispositivo de entra y salida para leer y escribir archivos de texto, binarios y recursos, puede ser utilizado por sí mismo o con un QTextStream o QDataStream.
- **QDebug:** proporciona un flujo de salida para la información de depuración, se utiliza cada vez que el desarrollador tiene que escribir la depuración o la información de seguimiento a un dispositivo, archivo, cadena o una consola.
- **QAction:** proporciona una acción abstracta de interfaz de usuario que se puede insertar en los widgets.
- **QObject:** es la clase base de todos los objetos de Qt, es el corazón del modelo de objetos de Qt. La característica central de este modelo es un mecanismo muy poderoso para la comunicación sin fisuras objeto denominado señales y slots.
- **QVariant:** posibilita una unión de los más comunes tipos de datos de Qt en un mismo entorno.
- **QTime:** proporciona funciones de reloj de tiempo, contiene un reloj de tiempo. Se puede leer la hora actual del reloj del sistema y medir un lapso de tiempo transcurrido. Se proporciona funciones para comparar los tiempos y para la manipulación de un tiempo mediante la adición de un número de milisegundos.
- **QMessageBox:** proporciona un cuadro de diálogo modal para informar al usuario o para pedir al usuario una pregunta y recibir una respuesta.
- **QThread:** proporciona hilos independientes del sistema, comparte datos con todos los otros hilos dentro del proceso, pero se ejecuta de forma independiente en la forma en que un programa separado hace en un sistema operativo multitarea. En lugar de comenzar en main (), QThreads comienza a ejecutarse en el run ().

- **QProcess:** se utiliza para iniciar los programas externos con una comunicación con ellos.

3.1.2 Otras bibliotecas

A pesar de ser Qt un IDE con excelentes facilidades de trabajo fue necesario utilizar otras bibliotecas para el desarrollo de la presente investigación.

MediaInfo

MediaInfo en una biblioteca compatible con Qt y el lenguaje C++ que posibilita obtener las propiedades y metadatos tanto de un audio como de un video, en esta investigación solo se trabajará con las propiedades de los audios (MundoDivX, 2012).

FFmpeg

FFmpeg es una biblioteca completa que se utiliza en el procesamiento tanto de audio como de video. Es desarrollado bajo Linux pero puede ser usado en la mayoría de los sistemas operativos. Es compatible con Qt que utiliza C++ (FFmpeg, 2012).

3.2 Diseño

La utilización de la metodología de desarrollo de software XP trae consigo muchas ventajas en el desenlace de la investigación ya que evita la generación innecesaria de documentación que caduca en el tiempo por la poca utilización que tiene dentro de un proyecto real. XP propone utilizar los diagramas UML para representar los distintos componentes del sistema que se desarrolla, pero no aclara cuáles se deben emplear, por tal motivo para el desarrollo de la presente investigación se decide utilizar las Tarjetas CRC para modelar la implementación del componente. El diseño con XP debe ser entendible, limpio, sencillo e incremental, de forma que se agreguen las funcionalidades a medida que la investigación va desarrollándose.

3.2.1 Tarjetas CRC

Como parte de la metodología XP las Tarjetas CRC posibilitan realizar el diseño del software orientado a objetos. Constituyen un puente de comunicación entre los participantes del entorno de trabajo y

colaboración. Las siglas CRC se corresponden con la función que realizan en el diseño de software (Clase, Responsabilidad y Colaboración). Permiten representar las clases implementadas como fuente de datos dando la posibilidad de conocer el comportamiento de cada una de ellas en un alto nivel y la representación del significado de las mismas (Beck, y otros, 1989).

Las tarjetas CRC están descritas por 3 partes fundamentales:

- **Nombre de la clase:** Se especifica el nombre de la clase que se describe.
- **Responsabilidades:** Son las funciones que implementa la clase.
- **Colaboradores:** Representa las demás clases con las que trabaja para llevar cumplir con sus responsabilidades.

Las Tarjetas CRC realizadas se referencian en el *Anexo # 3* de la investigación. A continuación se muestra un ejemplo de estas como resultado de la implementación de los plugins:

Class_MediaInfo	
Responsabilidades	**Colaboraciones**
Obtener Características Devolver lista de características Dar estado de proceso Dar ID del proceso Cancelar proceso	

Tabla 5: Tarjeta CRC Class_MediaInfo.

Estas son las principales clases que controlan el funcionamiento de los plugins. Algunas de ellas se relacionan entre sí con la posibilidad de la reutilización de código sin tener que volver a implementar funciones ya definidas, lo que constituye un coste menor de trabajo y tiempo y por lo tanto una ventaja para los sistemas de información.

3.3 Implementación

La implementación es la etapa donde los desarrolladores comienzan a definir el código fuente de la aplicación, donde se realizan cada una de las funcionalidades identificadas en un inicio y se comprueba que estas cumplas con las particularidades que se definieron.

Al utilizar XP se debe probar antes de comenzar a implementar, debido a la aplicación de *Test Driven Development* (del inglés TDD): Se basa en escribir una prueba, a continuación, se escribe el código suficiente para pasarla y después se mejora el código, principalmente para asegurar la legibilidad y eliminar duplicaciones. Con la aplicación de este proceso las pruebas se incorporan antes de tiempo al proceso de desarrollo de software. La implementación y las pruebas se realizan de manera simultánea, no se espera la culminación de la implementación para probar. Es un ciclo que consta de varios pasos para que funcione eficientemente (Wake, 2000):

- Escribe una prueba.
- Compila la prueba. Debe de fallar inicialmente porque la funcionalidad no se ha implementado aún.
- Implementa sólo lo suficiente para que compile.
- Corre la prueba y mírala fallar.
- Implementa sólo lo necesario para pasar la prueba.
- Corre la prueba y mírala funcionar.
- Refactoriza el código para una mejor claridad y elimina duplicaciones.
- Repite el proceso desde el principio.

3.3.1 1era Iteración

Para la realización de la 1era iteración se seleccionaron las historias de usuario 1, 2, 3 y 4, las cuales se desarrollaron en un período de 21 días. Las tareas realizadas se referencian en el *Anexo # 4* de la investigación. A continuación se presenta un ejemplo de estas tareas:

Tarea de ingeniería	
Número de tarea: 1	**No. De la HU:** 1
Nombre de la tarea: Extraer características de un archivo de audio.	
Tipo de tarea: Desarrollo	**Estimación:** 7 días
Fecha inicio: 05/03/2012	**Fecha fin:** 12/03/2012
Programador responsable: Janet Cristina Labrada Paneque	
Descripción: Esta tarea consiste en extraerle a un archivo de audio todas las características que el sistema de información requiera en su utilización.	

Tabla 6: Tarea # 1 de la HU # 1.

3.3.2 2da Iteración

Para la realización de la 2da iteración se seleccionaron las historias de usuario 4, 5, 6 y 7, las cuales se desarrollaron en un período de 21 días. Las tareas realizadas se referencian en el *Anexo # 5* de la investigación. A continuación se presenta un ejemplo de estas tareas:

Tarea de ingeniería	
Número de tarea: 2	**No. De la HU:** 4
Nombre de la tarea: Confeccionar huella digital.	
Tipo de tarea: Desarrollo	**Estimación:** 10 días
Fecha inicio: 02/04/2012	**Fecha fin:** 13/04/2012

Programador responsable: Janet Cristina Labrada Paneque
Descripción: Esta tarea consiste en completar la implementación de los algoritmos que posibilitarán la confección de la huella digital al archivo de audio que se manipula.

Tabla 7: Tarea # 2 de la HU # 4.

3.4 Pruebas

La metodología de desarrollo XP incorpora las pruebas tempranamente, gracias a aplicar TDD. El desarrollo dirigido por pruebas permite que se prueben constantemente los componentes más pequeños del código fuente: las clases, estas pruebas son realizadas por los desarrolladores y forman parte de las llamadas pruebas de unidad, que constituyen el nivel más bajo de las pruebas que se le aplican al software, como se llevan a cabo directamente sobre el código fuente pertenecen a las llamadas pruebas de caja blanca (Urbino, 2010).

En el segundo nivel de pruebas se encuentran las pruebas de integración. Estas pruebas se enfocan en encontrar errores en las interfaces de los componentes y en asegurar que todos los componentes operen correctamente. El tercer nivel corresponde a las pruebas del sistema. Las pruebas del sistema se ejecutan cuando el software funciona como un todo y todos los componentes de software y hardware han sido integrados. Por último, se encuentran las pruebas de aceptación, las cuales se realizan para asegurar que el software está listo para ser desplegado, dichas pruebas las realizan los usuarios finales.

Las pruebas más importantes que se aplican en un proceso de desarrollo basado en la metodología XP son las pruebas de unidad, al aplicar TDD y las pruebas de aceptación, las cuales constituyen acuerdos a los que llegan los clientes con los desarrolladores, un contrato que asegura que las funcionalidades pactadas han sido correctamente implementadas. Las pruebas de aceptación no se realizan solamente al final cuando el software está completo, sino que entre iteraciones se prueben todas las HU que se implementaron en la iteración anterior. Estas pruebas se documentan antes de implementar las HU en una iteración. Este proceso permite que la corrección de los errores detectados al probar el software durante y después de una iteración pueda agregarse como nuevas HU a incorporar en la siguiente iteración. Los casos de prueba de aceptación se referencian en el *Anexo # 6* de la presente investigación. A continuación se muestra un ejemplo de las pruebas de aceptación realizadas:

Caso de Prueba de Aceptación	
Código: HU_1_PA1	**Historia de Usuario: 1**
Nombre: Extraer características de los archivos de audio	
Descripción: Se extraen todas las características que el sistema desea conocer de un archivo de audio.	
Condiciones de ejecución: Biblioteca MediaInfo instalada en el sistema.	
Entrada/Pasos de ejecución: El sistema emite una señal que indica la ejecución del plugin que permite la extracción de características y notifica los parámetros acústicos que deben ser extraídos.	
Resultado esperado: Se obtiene la información de las características extraídas del archivo de audio.	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 8: Caso de Prueba de Aceptación 1.

3.5 Experimentos realizados al componente

Para la realización de los experimentos se confeccionó una interfaz de prueba que se encargara de utilizar el componente y mostrar su funcionamiento. Dicha interfaz se creó con la utilización de las mismas herramientas que sirvieron para implementar la solución de la presente investigación. Esta interfaz da la posibilidad de extraer las características que posee un archivo de audio, codificar el audio a los parámetros deseados por el usuario del sistema con que interactúa y a su vez conocer si el archivo posee la calidad establecida por el sistema. Todas estas funciones se realizaban al cargar los plugins correspondientes que permitían obtener la respuesta o acción deseada.

Medidas de Eficiencia

Cuando se realiza reconocimiento digital de audio se produce un falso positivo una vez que se detecta algún duplicado de audio que no lo es y un falso negativo cuando se concluye que no hay duplicado de audio mientras sí existe (Boqué, 2008). Ambas clasificaciones permiten calcular la eficacia del sistema.

Análogamente se definen los términos verdadero positivo y verdadero negativo para describir las decisiones correctas de cualquier proceso. Un verdadero positivo se produce cuando se detecta un duplicado de audio correctamente. Por otra parte, cuando se detecta que no hay duplicado y se confirma que no lo hay se está en presencia de un verdadero negativo. Para un mejor entendimiento de estas clasificaciones se presenta la siguiente tabla (Cruz, 2011):

Condición de duplicado de audio		**Duplicado**	**No Duplicado**
Resultado obtenido	**Positivo**	Duplicado + Positivo = **Verdadero Positivo**	No Duplicado + Positivo = **Falso Positivo**
	Negativo	Duplicado + Negativo = **Falso Negativo**	No duplicado + negativo = **Verdadero Negativo**

Tabla 9: Términos usados para describir los resultados del procesamiento de audio.

La eficiencia máxima se consigue cuando el sistema es capaz de procesar un conjunto amplio de audios sin producir ningún falso positivo o falso negativo. Esto se traduce en que los duplicados de audio detectados son correctos en un 100% y no debe quedar ningún duplicado sin detectar.

Para medir la calidad integral del audio se utilizan varias clasificaciones relacionadas con la percepción humana del sonido, las cuales se presentan en la Tabla 3 y se definen para el presente trabajo (Cruz, 2011):

- Excelente: Esta clasificación se detecta cuando el audio se percibe con muy buena intensidad, con una gran potencia acústica y se detectan muy claramente las diferencias entre los sonidos graves y los agudos. Se pueden distinguir notas musicales tocadas con diferentes instrumentos en la canción. No se percibe ruido o distorsión.

- Buena: Se pone de manifiesto cuando se detectan con claridad los sonidos graves y agudos, se mantiene una buena intensidad y no se percibe ruido o distorsión. La potencia acústica es buena.

- Regular: En ocasiones se percibe ruido y distorsión. Se pueden distinguir sonidos graves y agudos. La intensidad del sonido varía con frecuencia.

- Mala: Los archivos con esta clasificación presentan ruido y distorsión que hacen muy inestable la percepción del timbre y la altura en la canción. Cuesta trabajo distinguir notas musicales tocadas con los diferentes instrumentos.

- Muy mala: Solo se percibe mucho ruido y distorsión en la canción. No se encuentra diferencia entre lo grave o agudo que pueden ser los sonidos.

Experimentos

Para la verificación del componente realizado se decidió realizar una serie de experimentos que demostraran el correcto funcionamiento del mismo. Para validad la comprobación de la calidad sobre un archivo de audio se escogieron 120 archivos de audio que se utilizan en la plataforma PRIMICIA y se compararon con el patrón de calidad establecido por el usuario del sistema. Luego de comprobar si tenían calidad, se codificaron todos aquellos archivos de audio que no estaban dentro del patrón de calidad establecido.

Los archivos de audio seleccionados poseen diferentes parámetros que da una muestra de que no existe estandarización entre ellos. Sus frecuencias de muestreo poseen disímiles valores. Para poner en práctica el experimento deseado sobre estos archivos se escogió a una persona que realizara la función de oyente con el fin de detectar los cambios en los archivos al ser codificados.

Las frecuencias que se utilizaron para las conversiones fueron: 32 KHz, 36 KHz, 40 KHz, 48 KHz y 52 KHz debido a que la investigación precedente demostró la efectividad de las frecuencias por encima de 32 KHz para oyentes humanos. La tasa de bits y el número de canales no fue necesario probarlos mediante experimentos porque los valores recomendables para estas características se han justificado a lo largo de la investigación.

3.4 Resultados

Validación de la calidad de los archivos de audio

En el experimento realizado se analizaron archivos de audio que no se encontraban dentro de los parámetros de calidad definidos. Para demostrar la efectividad del componente realizado se codificaron estos archivos que no poseían calidad logrando obtener un archivo con las características deseadas.

Para el experimento de los 120 archivos de audio utilizados se pudo detectar utilizando el plugin Extraer Características que 72 tenían una frecuencia de 16 KHz, 25 de 24 Kz, 17 de 32 KHz y 6 de 36 KHz. Al utilizar el plugin Comprobar Calidad de demostró como estos archivos no cumplían con los parámetros de calidad establecidos por el sistema y se decidió realizar codificación digital sobre ellos. Luego de haber realizado todas estas transformaciones sobre los audios sin calidad se registraron las apreciaciones obtenidas por cada frecuencia.

	Frecuencia de muestreo (KHz)							
Clasificación de percepción	16	44.1	24	44.1	32	44.1	36	44.1
Excelente	0	68	3	23	7	17	2	6
Buena	13	4	14	2	5	0	3	0
Regular	17	0	6	0	2	0	1	0
Mala	28	0	2	0	2	0	0	0
Muy Mala	14	0	0	0	1	0	0	0

Tabla 10: Comparación entre frecuencias.

En frecuencias como 16 KHz predominaron las canciones con clasificaciones de percepción mala y muy mala, mientras para las mayores e iguales que 32 KHz los resultados muestran lo contrario, lo que es evidente al codificar los archivos de audio a una frecuencia de 44.1 KHz los mismos se obtienen con total nitidez y claridad en la reproducir. A una frecuencia de 32 KHz es posible captar la información ofrecida

por la media que se reproduce, pero la investigación ha demostrado que existe una notable mejoría cuando el audio posee un valor de frecuencia de 44.1 KHz.

3.4.2 Validación de la detección de duplicados

En el experimento realizado se analizaron archivos de audio repetidos que poseían metadatos diferentes de forma tal que fuera el plugin Detectar Duplicados el encargado de informar que el archivo ya se encontraba registrado en la base de datos utilizada para el almacenamiento de las medias. Para el experimento de los 120 archivos de audio utilizados 68 se encontraban duplicados. Para evaluar los resultados de este experimento se utilizaron los conceptos relacionados en la Tabla 9. A continuación se refleja el resultado obtenido luego de haber puesto en práctica las funcionalidades del componente para detectar duplicados de audio:

Resultados			
Verdaderos Positivos	**Falsos Positivos**	**Verdaderos Negativos**	**Falsos Negativos**
68	0	52	0

Tabla 11: Resultados de las pruebas teniendo en cuenta los audios analizados.

Como se pudo apreciar la detección de archivos duplicados arrojó resultados satisfactorios siendo detectados todos los archivos que se encontraban repetidos sin mostrar un resultado incorrecto. Este experimento demuestra la efectividad del plugin Detectar Duplicados y de la etapa de reconocimiento que sobre los archivos se efectúa.

3.6 Conclusiones parciales

En el presente capítulo se abordaron los temas que hacen alusión a la implementación de la solución y las pruebas a la solución final. El uso de las tarjetas CRC evidencia el comportamiento del componente de procesamiento de audio que se desarrolla, las clases que comprenden y las responsabilidades que ejecuta. Las tareas realizadas en cada una de las iteraciones, permitieron brindar una solución real con el desarrollo final del componente de audio. El desarrollo basado en pruebas apoyado por la realización de las pruebas de aceptación muestra resultados satisfactorios sobre el componente desarrollado y evidenció que es una estrategia fiable para obtener un producto probado y funcional.

CONCLUSIONES GENERALES

Al término de la presente investigación se obtuvo una propuesta de solución que da por cumplido el objetivo general planteado, con la realización de un componente de procesamiento digital de audio que agrupa un conjunto de plugins encargados de darle solución a la situación problemática abordada. De esta manera se puede arribar a las siguientes conclusiones:

- El uso de la metodología de desarrollo XP demuestra su efectividad en proyectos pequeños y con una interacción permanente con el cliente.
- El patrón arquitectónico Microkernel, con el uso de Observer y Reflection, es efectivo en aplicaciones orientadas a componentes.
- El componente desarrollado es flexible y escalable, lo que se evidencia con la adaptación de los plugins que lo componen a cualquier sistema de información audiovisual.

RECOMENDACIONES

A partir de los resultados obtenidos con la realización de este trabajo investigativo y basándose en las experiencias acumuladas a lo largo del desarrollo del mismo, se proponen las siguientes recomendaciones para futuros trabajos:

- Profundizar en el estudio de algoritmos de detección de duplicados en señales de audio para incluirle a los sistemas de información nuevas formas de eliminación de archivos repetidos, para que garantice la escalabilidad del sistema.
- Seleccionar otras bibliotecas existentes para la manipulación de audio con el fin de obtener mejores resultados en el tratamiento de este tipo de archivo.
- Proponer la utilización del componente de procesamiento digital de audio en otros sistemas de información que manejen este tipo de media en la universidad o fuera de esta.

BIBLIOGRAFÍA CITADA

1. **Alfaro, Félix Murillo. 1999.** Herramientas CASE. *Instituto Nacional de Estadísticas e Informática.* [En línea] Noviembre 1999. [Citado el: Noviembre 25, 2011.] http://www.inei.gob.pe/. 875-99-OI-OTDETI-INEI.
2. **ALLAMANCHE, Eric, HERRE, Jürgen y HELLMUTH, Oliver. 2001.** *Content-based Identification of Audio Material Using MPEG-7 Low Description.* Germany : s.n., 2001.
3. **Alvarado, Diego. 2005.** *Efecto del enventanado en la obtención del espectro discreto de una señal.* San Cayetano Alto - Loja Ecuador : s.n., 2005. (Curtis, 2009)
4. **Beck, Kent y Cunningham, Ward. 1989.** A Laboratory For Teaching Object-Oriented Thinking . [En línea] 10 10, 1989. [Citado el: 5 30, 2012.] http://c2.com/doc/oopsla89/paper.html.
5. **Blanchete, Jasmin y Summerfield, Mark. 2008.** *C++ GUI Programming with Qt 4.* s.l. : Prentice Hall, 2008. 978-0132354165.
6. **Boqué, Ricard. 2008.** *El límite de detección de un método analítico.* Tarragona. España : Universitat Rovira i Virgili, 2008.
7. **Briceño, Edgar Armando Vega. 2008.** *Los sistemas de información y su importancia para las organizaciones y empresas.* Costa Rica : s.n., 2008. 06-2005.
8. **Buschmann, Frank, y otros. 1996.** *Pattern Oriented Software Architecture: A system of patterns.* 1996.
9. **Cáceres, Juan Pablo. 2007.** *Análisis Espectral 1: Transformada Corta de Fourier y Ventanas. .* California : s.n., 2007.
10. **Cano, Pedro. 2007.** *CONTENT-BASED AUDIO SEARCH: FROM FINGERPRINTING TO SEMANTIC AUDIO RETRIEVAL.* 2007.
11. **Canós, José H., Letelier, Patricio y Penadés, María Carmen.** *Metodologías Ágiles en el Desarrollo de Software.* Universidad Politécnica de Valencia : s.n.
12. **Carrasco, Miguel A. 2002.** *RECONOCIMIENTO BIOMÉTRICO DE AUDIO Y ROSTRO: UN SISTEMA.* Santiago de Chile : s.n., 2002.
13. **Cebrian, M. 1995.** *Información audiovisual. Concepto, técnica, expresión y aplicaciones.* Madrid : Síntesis. S.A, 1995.
14. **CloneSpy. 2012.** http://www.clonespy.com/. *http://www.blogtecnia.com/2008/11/clonespy-portable-elimina-archivos.html.* [En línea] 4 28, 2012. [Citado el: 5 30, 2012.] http://www.clonespy.com/.

15. **CNET. 2009.** CNET. *http://download.cnet.com/Duplicate-Cleaner/3000-2248_4-10584403.html.* [En línea] 5 13, 2009. [Citado el: 5 30, 2012.] http://www.cnet.com/.

16. **Cruz, Ariadna Miranda. 2011.** *Procesamiento Digital de Audio para la Plataforma de Televisión Informativa PRIMICIA.* Cuidad de la Habana : s.n., 2011. pág. 75, Tesis.

17. **Curtis, Keith. 2009.** *After the Software Wars.* Seattle : s.n., 2009.

18. **Eclipse, Fundación. 2012.** Eclipse. [En línea] The Eclipse Foundation, 2012. [Citado el: 5 30, 2012.] http://www.eclipse.org/.

19. **FFmpeg. 2012.** FFmpeg. [En línea] 2012. [Citado el: 5 30, 2012.] http://ffmpeg.org/.

20. **Foundation, Python Software. 2012.** python. [En línea] Python Software Foundation, 2012. [Citado el: 5 30, 2012.] http://www.python.org/.

21. **Friedrich, Guillermo R. 2002.** *Introducción al procesamiento digital de señales.* Bahía Blanca : s.n., 2002.

22. **Geekgt. 2009.** http://geekgt.com/. *http://www.bigbangenterprises.de/.* [En línea] 8 4, 2009. [Citado el: 5 30, 2012.] http://www.bigbangenterprises.de/en/doublekiller/.

23. **Goto, Masataka. 2003.** *A Chorus-Section Detecting Method For Musical Audio Signals.* Japan : Japan Science and technology Corporation, 2003.

24. **Heineman, G. y Councill, W. 2001.** *Component Based Software Engineering.* s.l. : Addison-Wesley, 2001.

25. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 1999.** *El proceso unificado de desarrollo de software.* Madrid : s.n., 1999.

26. **J.G.Proakis, D.G.Manolakis. 1996.** *Digital Signal Processing: Principles, algorithms and applications.* s.l. : Prentice-Hall, Inc, 1996.

27. **Kiyosaki, Robert y Lechter, Sharon. 1997.** *Padre Rico, Padre Pobre.* EE.UU : s.n., 1997. pág. 207. ISBN 0-446-67745-0.

28. **Kniberg, Henrik. 2007.** *Scrum y XP desde las trincheras.* s.l. : C4Media, 2007.

29. **Knowlton, Jim. 2009.** *Python.* Madrid : Anaya Multimedia, 2009. pág. 272. Vol. 1. 9788441525139.

30. **Laudon, Kenneth C. y Laudon, Jane P.** *Administración de los sistemas de información. Organización y tecnología.* s.l. : Prentice Hall.

31. **León, Rolando Alfredo Hernández y Coello González, Sayda. 2011.** *El proceso de investigación científica.* Ciudad de La Habana : Editorial Universitaria del Ministerio de Educación Superior, 2011. pág. 110. ISBN 978-959-16-1307-3.

32. *Los medios de enseñanza: clasificación, selección y aplicación.* **Bravo Ramos, Juan Luis. 2004.** 24, Madrid, España : s.n., 2004. ISSN 1133-8482.

33. **Maldonado, J. 2003.** *Automático de Señales de la Voz Venezolana.* Venezuela : s.n., 2003.

34. **Martínez, Luz García. 2007.** *Ecualización de histogramas en el procesado robusto de voz.* Granada : s.n., 2007. 978-84-338-4698-3.

35. *Metodología de desarrollo del software.* **Pérez, Isaías Carrillo, Pérez González, Rodrigo y Rodríguez Martín, Aureliano David. 2008.** 2008.

36. **Milián, David Luis García. 2011.** *Implementación del Módulo de Gestión de Medias de la Plataforma de Televisión Informativa Primicia v 2.0.* Cuidad de la Habana : s.n., 2011.

37. **Moles, A. 1982.** *La comunicación y los mass media.* Bilbao : Mensajero, 1982. pág. 341.

38. **Montilva, Jonás A., Arapés, Nelson y Colmenares, Juan Andrés. 2003.** *Desarrollo de software basado en componentes.* Mérida, Venezuela : s.n., 2003.

39. **MundoDivX. 2012.** MundoDivX. [En línea] 2012. [Citado el: 5 30, 2012.] http://www.mundodivx.com/codecs/mediainfo.php.

40. **Muñoz, León Alberto Martínez. 2011.** *Implementación de un editor gráfico de circuitos.* Cartagena. Colombia : Universidad Politécnica de Cartagena, 2011.

41. **Neira, Elkin Ramón Garavito. 2010.** *Modelo de identificación de locutor en entornos GSM, aplicación en Colombia.* Bogotá, D.C. Colombia : s.n., 2010.

42. **NoClone. 2007.** NoClone. [En línea] 9 28, 2007. [Citado el: 5 30, 2012.] http://classic.noclone.net/.

43. **Nokia Corporation. 2011.** Qt Developer Network. [En línea] Nokia, 2011. [Citado el: 5 30, 2012.] http://doc.qt.nokia.com/4.7-snapshot/assistant-manual.html.

44. **Ochoa, Sergio. 2005.** Introducción a los patrones (diseño y arquitectura). La Habana : s.n., 2005.

45. **Oppenheim, Alan V. 1998.** *Señales y sistemas.* s.l. : Pearson, 1998. ISBN 970-17-0116-X.

46. **Ramiro, Daniel Isaac Khan. 2011.** *Interfaz gráfica multiplataforma para la simulación de ecuaciones físicas.* Madrid : Universidad Rey Juan Carlos, 2011.

47. **Reverón, Antonio Fumero. 2011.** *Medios para la información, la relación y la comunicación en la web 2.0.* Madrid, España : s.n., 2011.

48. **Rojas, Maribel Ariza y Molina García, Juan Carlos. 2004.** *Introducción y principios básicos del desarrollo de software basado en componentes.* Bogotá : s.n., 2004.

49. Rosales, Juan Camacho. 2010. *Estadística con SPSS para Windows.* Madrid : s.n., 2010. 970-15-0585-9.

50. **Santos, William de la Cruz De los y Altamirano, Leopoldo. 2009.** *Programación paralela y concurretnte en C++.* Juárez. México : s.n., 2009.

51. **Sametinger, J. 1997.** *Software engineering with reusable components.* s.l. : Springer Verlag, 1997.

52. **Saráchaga, Gabriela, Sartori, Virginia y Vignoli, Laura. 2006.** *Identificación automática de resumen en canciones.* 2006.

53. **Sierra, María. 2006.** *Trabajando con Visual Paradigm for UML.* Cantabria : s.n., 2006.

54. **Sommerville, Ian. 2000.** *Software engineering.* 6ta. s.l. : Addison-Wesley Pub Co, 2000.

55. **Stroustrup, Bjarne. 1998.** *El lenguaje de programación C++.* Madrid : Addison Wesley, 1998. ISBN 84-7829-019-2.

56. **Urbino, Rafael Jacobo Hidalgo. 2010.** *Análisis, diseño e implementación de una herramienta que permita la centralización de la información gestionada por el módulo Resultados de la Colección Multisaber".* Cuidad de la Habana : s.n., 2010.

57. **Wake, William C. 2000.** *Extreme Programming Explored.* 2000.

GLOSARIO DE TÉRMINOS

***Bitrate* o tasa de bits:** Número de bits que se transmiten por unidad de tiempo a través de un sistema de transmisión digital o entre dos dispositivos digitales. Es la velocidad de transferencia de datos.

Códec: Algoritmo utilizado para la compresión de datos. Es la abreviatura de codificador-decodificador. Los códecs pueden codificar el flujo o la señal (a menudo para la transmisión, el almacenaje o el cifrado) y recuperarlo o descifrarlo del mismo modo para la reproducción o la manipulación en un formato más apropiado para estas operaciones.

Enventanado: Aplicación (multiplicación) sobre la señal de voz completa de una función limitada en el tiempo (ventana), que produce una nueva señal de voz, cuyo valor fuera del intervalo definido por la ventana es nulo.

Framework: es un conjunto de bibliotecas o clases que pueden ser empleadas por los programadores para reutilizar su código en sus programas.

Frecuencia: Término empleado en física para indicar el número de veces que se repite en un segundo cualquier fenómeno periódico. La frecuencia es muy importante en muchas áreas de la física, como la mecánica o el estudio de las ondas de sonido.

Herramienta CASE: es un software que permite a los desarrolladores modelar parte o todos los componentes de una aplicación. Generalmente a través de diagramas. Las siglas CASE provienen de *Computer Aided Software Engineering*, que en castellano significan Ingeniería de Software asistida por computadora.

Hertz (Hz): Unidad de medida de la frecuencia, el ciclo por segundo (hercio, Hz). Un kilohercio (KHz) es 1000 ciclos por segundo, un megahercio (MHz) es un millón de ciclos por segundo y un gigahercio (GHz), 1000 millones de ciclos por segundo.

IDE: del inglés *Integrated Development Environment*, en español Entorno de Desarrollo Integrado, es un programa que ofrece una serie de herramientas que facilitan el trabajo de los desarrolladores de software para programar sus programas.

Metodología de desarrollo de software: a grandes rasgos una metodología de desarrollo de software define en un proyecto **quién** debe hacer **qué** cosa, **cuándo** se le dará cumplimiento y **cómo** lo va a hacer.

SDK: (siglas en inglés de software development kit) es un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto, por ejemplo paquetes de software, frameworks, plataformas de hardware, computadoras, sistemas operativos, etc.

UML: Del inglés *Unified Modeling Language*, en castellano Lenguaje Unificado de Modelado es una de las herramientas más importantes en el mundo actual del desarrollo de sistemas. Es un lenguaje visual para especificar, construir y documentar los artefactos de los sistemas de software y es aplicado en sistemas orientados a objetos.