Universidad de las Ciencias Informáticas
FACULTAD 6

Título: Sistema Gestor de Procesos de Media v2

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Raydel Raúl Viñolo Sosa, Alexander Roquero Figueroa

Tutor: Ing. Adnan Fuentes Díaz

La Habana, 25 de junio de 2012

"Año 54 de la Revolución"

DECLARACIÓN DE AUTORÍA

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

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

Raydel Raúl Viñolo Sosa
Autor

Alexander Roquero Figueroa
Autor

Adnan Fuentes Díaz
Tutor

DATOS DE CONTACTO

DEDICATORIA

De Alexander:

Dedico este trabajo, en especial, a mis padres Dania y Alexis por ser ejemplares y los principales guías de mis acciones, por haber confiado siempre en mí y apoyarme en todas mis decisiones. A mi tía Natacha por ser una segunda madre. Lo dedico además a mis abuelos, Marcos y en especial a Manolito, que hubiera disfrutando mucho este momento y a mis abuelas, Raquel y María Esther por ser las fuentes de comprensión de la familia. A mi hermana Yaima y a mi prima Arianna que me enseñan a ser mejor con sus críticas y cariños. A mi novia Delis por su apoyo incondicional en cada momento de la vida, por ser la promotora de buenos momentos y la portadora de la mayor alegría del mundo y a su familia, por la gran confianza depositada en mí.

De Raydel:

Este trabajo está dedicado a mis padres Susana y Raúl, que nunca me han defraudado, han sido mi guía y ferviente ejemplo de esfuerzo, empeño y sacrificio en la vida, por ellos y por su enorme amor me he convertido en el hombre que soy ahora. A mis abuelas Aurora y María por el amor infinito que siempre me han dado. A mis queridos hermanos Raudel y Francisco, los mejores hermanos del mundo. A mí adorada prima Maité y a mí querida sobrina Suleiky, siempre están presentes en mi corazón. A mi novia Maidy (Poti) por todo el amor que me da y por lo especial que me hace sentir cuando estoy a su lado.

AGRADECIMIENTOS

De Alexander:

Agradezco a toda mi familia que han hecho de mi lo que soy, a mis padres que son el pilar de mi vida, a mi hermana, mi tía Natacha y a mis abuelos por el apoyo incondicional. A mis tíos, en especial a Damaris, Nelson y Riqui que han estado cerca a pesar de la distancia. A mis compañeros de aula y del apartamento por estos cinco años de alegrías y tensiones. A mi tutor muy especialmente, por el empeño puesto en este trabajo, por la confianza, por su incondicionalidad y por ser un eslabón fundamental en el desarrollo de este. Y sobre todo a mi novia por ser la mayor alegría de mi vida, junto a mi familia.

De Raydel:

A mis padres por enseñarme a proteger y a luchar por lo que se ama, por su amor, confianza, apoyo incondicional a mis decisiones y por enseñarme que con empeño y esfuerzo todo se puede lograr. Gracias a mis abuelas, hermanos y a toda mi familia por su amor y por ayudarme siempre en todo lo que les fue posible, se que lo saben pero de igual forma se los digo: los quiero con el alma y no hay un día de este mundo que no estén en mi corazón. A Maidy por estar siempre a mi lado, brindarme su mano tierna y ayudarme en todo lo que necesite.

A todos mis amigos y compañeros que de una forma u otra han convivido conmigo durante estos cinco años y considero que han sido una segunda familia, en especial a Emilio y a Maidelyn por ser dos hermanos más, a Leyanis, Juan Pablo, Dayana, Yoralis, Dunia, Viana, Sucel, Anisleidis, Janet Cristina, Yelenis, Analay, Miriam, Aliuzka, Ariel Labrada, Jose Carlos, Odilen, Josué, Liuben, Orlando, Walfrido, Yasmany Palmero, Yandys, Yosvany, Lambert, Julio, Reiner, Rojas, Arcides, Luis Alberto, Yordanis Carralero, Daril, Enrique, El Doble, Carlos. A mi tutor Adnan por guiarme y apoyarme en esta última faena de la carrera, gracias por la confianza, el interés y la dedicación. A mi equipo de trabajo, por ser otra familia en la cual confiar y con la cual pude contar para todo lo que necesité en estos años.

RECONOCIMIENTO

El presente trabajo se presentó en la Jornada Científica Estudiantil 2012 de la facultad #6 obteniendo la máxima calificación de Relevante. Se presentó además en la Jornada Científica Estudiantil UCI 2012 donde alcanzó de igual forma la calificación de Relevante.

RESUMEN

En la Universidad de las Ciencias Informáticas (UCI), específicamente en el departamento de Señales Digitales, han desarrollado una herramienta nombrada Sistema Gestor de Procesos de Media, su objetivo es automatizar procedimientos que se hacen de forma manual actualmente en muchas Televisoras. Esta herramienta no posee las características necesarias que permitan realizar personalizaciones de software para satisfacer las necesidades de los clientes en breves espacios de tiempo. Este documento recoge los resultados del proceso de desarrollo de una nueva versión de este software.

El Sistema Gestor de Procesos de Media v2 está compuesto por dos módulos el Gestor de Procesos de Media y la Plataforma de Codificación e Indexación. Para la elaboración de la solución que se expone en el presente trabajo se utilizó una arquitectura Orientada a Servicios y Basada en Componentes garantizando una arquitectura adaptable a diferentes entornos, característica muy reclamada actualmente a nivel mundial en este tipo de sistemas. Se usó RUP como metodología de desarrollo, C ++ como lenguaje de programación y Qt Creator como IDE de desarrollo. Todas las herramientas y tecnologías que fueron empleadas poseen licencias completamente libres, garantizando la independencia de licencias propietarias. El Sistema Gestor de Procesos de Media v2 permite la integración de sistemas con sistemas y sistemas con equipamiento de hardware, además de ser integrable con ambientes ya existentes en distintos negocios.

PALABRAS CLAVES

Gestión, proceso, media, componente, servicio, nodos.

ABSTRACT

At the University of Informatics Sciences (UCI), specifically the Digital Signal department, have developed a tool named Process Management System Media, its goal is to automate procedures that are currently manually in many Television Stations. This tool does not have the necessary characteristics that allow for customization of software to meet customer needs in a short space of time. This document contains the results of the process of developing a new version of this software. System Process Manager v2 Media consists of two modules of the Process Manager and Platform Media Coding and Indexing. To prepare the solution presented in this work we used a service-oriented architecture and component-based architecture ensuring adaptable to different environments, very unclaimed property currently worldwide in such systems. RUP was used as a development methodology, C + + as programming language and IDE Qt Creator and development. All tools and technologies that were used have completely free licenses, ensuring the independence of proprietary licenses. System Process Manager v2 Media enables integration of systems with systems and systems with hardware equipment as well as being integrated with existing environments in different businesses.

KEYWORDS

Management, process, media, component, service, nodes.

ÍNDICE DE FIGURAS

Figura 1 Disciplinas, Fases e Iteraciones de RUP) (Vallespir, 2011). ..... - 20 -
Figura 2 Diagrama de clases del diseño del Gestor de Procesos de Media. ..... - 40 -
Figura 3 Diagrama de clases del diseño de la Plataforma de Codificación..... - 41 -
Figura 4 Diagrama de Clases del Diseño de Plugins Lógica ..... - 42 -
Figura 5 Diagrama de Clases del Diseño de Plugins Ejecución. ..... - 42 -
Figura 6 Nodo Plataforma de Codificación. ..... - 46 -
Figura 7 Nodo Esclavo Tercero. ..... - 47 -
Figura 8 Clientes Externos. ..... - 47 -
Figura 9 Diagrama de Componentes de Sistema Gestor de Procesos de Media v2. ..... - 48 -

ÍNDICE

DEDICATORIA I
AGRADECIMIENTOS II
RECONOCIMIENTO III
RESUMEN IV
ABSTRACT V
ÍNDICE DE FIGURAS VI
INTRODUCCIÓN ..... - 1 -
CAPÍTULO 1: Fundamentación teórica ..... - 6 -
1.1 Introducción ..... - 6 -
1.2 Conceptos asociados al dominio del problema ..... - 6 -
1.2.1 Gestión ..... - 6 -
1.2.2 Proceso ..... - 6 -
1.2.3 Media ..... - 6 -
1.2.4 Codificar ..... - 7 -
1.2.5 Indexación ..... - 7 -
1.2.6 Transferencia ..... - 7 -
1.2.7 Actualización de metadatos de media ..... - 7 -
1.2.8 Técnicas para balanceo de cargas operacionales ..... - 8 -
1.2.9 Persistencia de datos ..... - 8 -
1.2.10 Tolerancia a fallos ..... - 8 -
1.3 Objeto de Estudio ..... - 9 -
1.3.1 Descripción General ..... - 9 -
1.3.2 Descripción actual del dominio del problema ..... - 11 -
1.3.3 Situación Problemática ..... - 12 -
1.4 Análisis de otras soluciones existentes ..... - 13 -
1.4.1 TDIndexer ..... - 13 -
1.4.2 TDMPM ..... - 15 -
1.4.3 Pro Media Carbon ..... - 16 -
1.5 Conclusiones Parciales ..... - 17 -
CAPÍTULO 2: Herramientas y Tecnologías a utilizar ..... - 18 -
2.1 Introducción ..... - 18 -
2.2 Metodología de desarrollo de software ..... - 18 -
2.2.1 RUP como metodología a utilizar ..... - 18 -

2.3 UML - 20 -
2.4 Visual Paradigm como herramienta CASE - 21 -
2.5 Lenguaje de programación a utilizar - 22 -
2.6 Entorno Integrado de Desarrollo - 23 -
2.6.1 Qt Creator como IDE de desarrollo - 23 -
2.7 Sistema gestor de base de datos - 24 -
2.7.1 PostgreSQL y SQlite como sistemas gestores de base de datos - 24 -
2.7.2 SQLite - 25 -
2.8 XMLRPC como tecnología de comunicación - 25 -
2.9 BPEL - 26 -
2.10 Conclusiones Parciales - 27 -
CAPÍTULO 3: Descripción de la solución propuesta - 28 -
3.1 Introducción - 28 -
3.2 Patrones y Estilos de Diseño de Software - 28 -
3.2.1 Patrones de Diseño GOF empleados - 28 -
3.2.1.1 Patrones Creacionales - 29 -
3.2.1.2 Patrones Estructurales - 29 -
3.2.1.3 Patrones de Comportamiento - 30 -
3.2.2 Patrones Generales de Software para Asignar Responsabilidades (GRASP) - 30 -
3.2.2.1 Experto (*Expert*) - 31 -
3.2.2.2 Creador (*Creator*) - 31 -
3.2.2.3 Bajo Acoplamiento (*Low Coupling*) - 31 -
3.2.2.4 Alta Cohesión (*High Cohesion*) - 32 -
3.3 Patrones Arquitectónicos - 32 -
3.3.1 Arquitectura Pizarra o Repositorio - 32 -
3.3.2 Arquitectura en Capas - 33 -
3.3.3 Arquitectura Orientada a Objetos (Arquitectura OO) - 33 -
3.3.4 Arquitecturas Basadas en Componentes - 34 -
3.3.5 Arquitectura Orientada a Servicios - 35 -
3.4 Requisitos - 36 -
3.4.1 Requisitos funcionales de software - 36 -
3.4.2 Requisitos no funcionales de software - 39 -
3.5 Valoración crítica del diseño propuesto por el analista - 40 -
3.5.1 Módulo Gestor de Procesos de Media - 40 -
3.5.2 Módulo Plataforma de Codificación e Indexación - 41 -

3.5.2.1 Plugins Lógica - 42 -
3.5.2.2 Plugins Ejecución - 42 -
3.5.2.3 Plugins Consumir - 42 -
3.5.2.4 Plugins Publicar - 43 -
3.5.2.5 Útiles - 43 -
3.6 Diagrama Entidad-Relación del Gestor de Procesos de Media - 44 -
3.7 Modelo de Despliegue - 45 -
3.7.1 Descripción de componentes - 46 -
3.7.1.1 Nodo Gestor de Procesos de Media - 46 -
3.7.1.2 Nodo Plataforma de Codificación e Indexación - 46 -
3.7.1.3 Nodo Esclavo Tercero - 47 -
3.7.1.4 Clientes Externos - 47 -
3.8 Diagrama de componentes - 47 -
3.9 Estilos de Programación - 48 -
3.9.1 Estilo K&R y BSD KNF - 49 -
3.9.2 Estilo Allman - 50 -
3.9.3 Estilo GNU - 50 -
3.10 Estándar de Codificación - 51 -
3.11 Conclusiones Parciales - 52 -
CAPÍTULO 4: Validación de la solución propuesta - 53 -
4.1 Introducción - 53 -
4.2 Proceso de Pruebas - 53 -
4.2.1 Estrategia de Prueba del software - 54 -
4.2.1.1 Integración - 54 -
4.2.1.2 Integración Incremental Ascendente - 54 -
4.2.1.3 Integración entre Módulos del Sistema - 54 -
4.2.1.4 Integración con terceros - 55 -
4.2.2 Pruebas del Sistema - 56 -
4.2.3 Prueba de carga - 56 -
4.3 Conclusiones Parciales - 57 -
Conclusiones - 58 -
Recomendaciones - 59 -
Trabajos citados - 60 -
GLOSARIO DE TÉRMINOS - 63 -

INTRODUCCIÓN

Desde sus inicios el hombre tuvo la necesidad de transmitir información, primero lo hizo mediante señas, luego a través de la voz y posteriormente con la escritura. Con el transcurso del tiempo, el desarrollo de las ciudades y el aumento de las poblaciones, surge la necesidad de medios de comunicación masivos, dando lugar al surgimiento de los periódicos, la radio y posteriormente la televisión.

El surgimiento de la televisión tuvo gran relevancia, con ella las personas no solo percibían sonidos mediante la radio como se conocía hasta el momento, ya se podían observar a la par imágenes en movimiento que provenían de lugares muy distantes. Surge así la industria de la televisión y las conocidas televisoras, empresas dedicadas a la recepción, producción, almacenamiento y transmisión de materiales audiovisuales.

En la actualidad, la rama de la informática y las comunicaciones, producto del alto desarrollo que ha alcanzado y por las facilidades de automatización que proveen, ha sido incluida prácticamente en todas las esferas de la sociedad; la televisión no escapa a este fenómeno, a raíz de esto se han logrado avances significativos, pues las tareas que antes eran realizadas por personas actualmente son realizadas en menos tiempo y con más calidad por ordenadores en los que se ejecutan potentes sistemas informáticos. Estos sistemas necesitan ser lo suficientemente flexibles y estar actualizados tecnológicamente para ser funcionales ante las nuevas necesidades que surgen producto del incesante desarrollo del medio.

Cuba, inmersa en la industria del software, ha alcanzado un significativo nivel en los últimos años; un papel protagónico en este proceso lo ha desarrollado la Universidad de las Ciencias Informáticas (UCI) constituyendo actualmente una de las mayores productoras de software del país. Entre los centros productivos que posee la UCI se encuentra el Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED) y dentro de este el departamento de Señales Digitales, especializado en la universidad en el desarrollo de soluciones informáticas orientadas al campo de los audiovisuales.

Actualmente en el departamento de Señales Digitales existe un producto de software cuyo fin es la gestión de procesos de codificación, indexación, transferencia y actualización de metadatos de media. Los procesos de gestión surgen cuando desde aplicaciones externas se realizan peticiones al Gestor de Procesos de Media, este es el encargado de generar flujos de varias operaciones que serán

ejecutadas en distintos computadores dedicados o nodos esclavos como también son denotados. Esta solución se ve limitada a soportar nodos dedicados del tipo Plataforma de Codificación e Indexación. Si se necesitara gestionar otras operaciones como de almacenamiento, captura o transmisión, la arquitectura actual del sistema no es capaz de incluir otros nodos que permitan el procesamiento de estas, además de solo poder realizar los procesos antes mencionados y en un orden predefinido. Todo esto dificulta la posibilidad de realizar personalizaciones del sistema para diferentes negocios en cortos períodos de tiempo sin que esto involucre gasto de recursos materiales y de personal calificado que pudieran destinarse a otras actividades.

A partir de la problemática existente se plantea como **problema a resolver:** ¿Cómo reducir las limitaciones de integración y de dinamismo de la solución actual para la gestión de los procesos de codificación, indexación, transferencia de audiovisuales y de actualización de metadatos de media, existente en el departamento de Señales Digitales?

Se define como **objeto de estudio** los procesos de gestión de materiales audiovisuales.

Teniéndose como **objetivo general:** desarrollar un sistema que permita la gestión dinámica de los procesos de media en el departamento de Señales Digitales. El **campo de acción** se enmarca en la gestión dinámica de materiales audiovisuales en el departamento de Señales Digitales.

Se plantea como **Idea a defender** que**:** El desarrollo de un sistema que permita la gestión dinámica de los procesos de media, reduce las limitaciones de integración y de dinamismo de la solución actual para la gestión de los procesos de codificación, indexación, transferencia de audiovisuales y de actualización de metadatos de media, existente en el departamento de Señales Digitales.

Para dar cumplimiento al objetivo planteado se han identificado las siguientes **tareas de la investigación:**

1. Caracterizar las tecnologías que permiten mantener una arquitectura orientada a servicios.
2. Caracterizar las arquitecturas y modelos de programación grid.
3. Identificar la tecnología que permite la distribución de servicios orientada a arquitecturas grid.
4. Describir el funcionamiento de aplicaciones servidoras para la codificación e indexación de video.
5. Determinar técnicas de balanceo de cargas operacionales.
6. Caracterizar las arquitecturas de componentes adicionales.

7. Identificar el funcionamiento de estructura de programación dinámica.
8. Analizar las principales funcionalidades y servicios que brindan sistemas similares.
9. Realizar el Diseño del Sistema Gestor de Procesos de Media v2.
10. Implementar el Sistema Gestor de Procesos de Media v2.
11. Aplicar una estrategia de pruebas al Sistema Gestor de Procesos de Media v2.

Para desarrollar estas tareas se utilizan los siguientes métodos científicos:

Dentro de los métodos teóricos empleados se encuentran:

Analítico -Sintético: Se utiliza para el análisis de trabajos y documentos facilitando la obtención de conocimientos importantes para la investigación, permitiendo obtener los elementos más significativos que se relacionan con el objeto de estudio.

Histórico-Lógico: Se utiliza en el estudio y análisis bibliográfico de diferentes autores para poder realizar una amplia investigación sobre la evolución y desarrollo de los sistemas para la gestión de audiovisuales.

Modelación: Se utiliza para la creación de modelos que representan abstracciones con el objetivo de comprender el funcionamiento del proceso de gestión de medios audiovisuales.

Dentro de los métodos empíricos:

Observación: Se utiliza para visualizar el proceso de gestión de ficheros audiovisuales, cómo ocurre y su posible evolución. Permite analizar y obtener informaciones a partir de lo observado en relación al funcionamiento de sistemas similares, lo que da una visión de cómo tiene que ser el sistema a realizar.

Posibles resultados:

- Documentación que especifica el funcionamiento de tecnologías de software orientada a servicios.
- Documentación que recoge características de arquitecturas grid y modelos de programación en dichas arquitecturas.
- Documentación que identifica la tecnología para la distribución de servicios a utilizar así como su implantación de arquitecturas grid.

- Documentación que recoge el funcionamiento de aplicaciones encargadas de codificar e indexar video.
- Documentación que caracteriza e identifica la técnica de balanceo de carga a utilizar.
- Documentación que recoge el funcionamiento de las arquitecturas de componentes adicionales.
- Documentación que recoge la estructura de programación dinámica. Poner en práctica mediante ejemplos.
- Documentación que refleja el análisis de metodologías y métodos de funcionamiento y servicios brindados por aplicaciones similares.
- Documentación UML de los artefactos resultantes del análisis y diseño del Sistema Gestor de Procesos de Media.

Artefactos:

- Clases del Diseño con las descripciones de los atributos y métodos.
- Modelo de Implementación: diagrama de componentes.
- Ejemplo de código fuente de las principales clases.
- Plan de integración con los restantes módulos del sistema.
- Software, Sistema Gestor de Procesos de Media v2.
- Informe de pruebas realizadas al Sistema Gestor de Procesos de Media v2.

La investigación se estructura de la siguiente forma:

Capítulo 1: Se abordan los conceptos y definiciones de modelos y patrones de desarrollo de software asociados al objeto de estudio. Se caracteriza la versión actual del sistema Gestor de Procesos de Media, la situación existente en el departamento de Señales Digitales y las soluciones similares a nivel mundial.

Capítulo 2: Se especifican las principales herramientas, tecnologías, metodologías y lenguajes de programación que se utilizarán en la construcción de la solución para ofrecer una solución al problema científico de la investigación y complementar el objetivo general.

Capítulo 3: Se especifican conceptos relacionados con las definiciones de estilo y patrones de diseño de software, estilos de programación, estándar de codificación empleado. Se caracteriza la solución propuesta y se valora el diseño de clases propuesto por el analista.

Capítulo 4: Se exponen los resultados de la validación de la solución propuesta y se especifican los artefactos referidos a las etapas de pruebas.

CAPÍTULO 1: Fundamentación teórica

1.1 Introducción

El presente capítulo, para una mejor comprensión del problema, tiene como objetivos principales, los conceptos y definiciones asociados al objeto de estudio, al dominio del problema planteado y exponer una visión actualizada del estado de las principales herramientas similares que compiten hoy en el mercado mundial.

1.2 Conceptos asociados al dominio del problema

1.2.1 Gestión

Del latín *gestĭo*, el concepto de gestión hace referencia a la acción y al efecto de gestionar o de administrar. Gestionar es realizar diligencias conducentes al logro de un negocio o de un deseo cualquiera. El término gestión, por lo tanto, implica al conjunto de trámites que se llevan a cabo para resolver un asunto o concretar un proyecto. La gestión es también la dirección o administración de una empresa o de un negocio (definicion.de, 2008). Acción y efecto de hacer diligencias conducentes al logro de un negocio o de un deseo cualquiera. Proceso de planear, organizar, dirigir, evaluar y controlar (Restrepo, 2011). La gestión es la actividad o las actividades que se realizan para organizar, dirigir y administrar medios en aras de lograr resultados satisfactorios, rentables y productivos.

1.2.2 Proceso

Proceso es el conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados (Muro, 2010). Se denomina proceso al conjunto de acciones o actividades sistematizadas que se realizan o tienen lugar con un fin. En la informática, un proceso puede querer decir distintas combinaciones operativas que ocurren simultáneamente para alcanzar un resultado o un producto (David, 2008). Acción de ir hacia adelante, conjunto de las fases sucesivas de un fenómeno natural o de una operación artificial (ESPAÑOLA, 2011). Proceso es el conjunto de tareas o actividades sistematizadas que de forma sincronizada, opcional o paralelamente transforman elementos y partes en un efecto o utilidad.

1.2.3 Media

En la presente investigación el término **media** se refiere a ficheros de imagen, audio o video que

pueden ser almacenados, reproducidos, transmitidos, codificados, indexados o eliminados.

1.2.4 Codificar

Codificar operación encargada de representar una información mediante un código, por ejemplo, representar cada carácter alfanumérico mediante de un conjunto de bits valor 0 a 1 (Definiciones, 2007).

Orientado a la solución propuesta en la investigación, codificar representa la acción de convertir mediante el empleo de un software especializado una media de un formato a otro, ejemplo mkv, h264, mpg y mp3.

1.2.5 Indexación

Indexación según la Real Academia Española (RAE) es la acción de hacer índices o registrar ordenadamente datos e informaciones (ESPAÑOLA, 2011). La presente investigación enmarca los procesamientos realizados sobre medias audio visuales con el objetivo de agregarle o extraer algún tipo de información, ejemplo extracción de fotogramas, extracción e incorporación de subtítulo, extracción de audio, segmentación y reconocimiento de patrones.

1.2.6 Transferencia

Del latín *transferens* RAE define transferencia como pasar o llevar algo desde un lugar a otro (ESPAÑOLA, 2011). Se define transferencia al proceso de trasladar una media desde un dispositivo de almacenamiento hasta otro. Las transferencias se originan producto de la necesidad de almacenar la media resultan te de un procesamiento.

1.2.7 Actualización de metadatos de media

Los metadatos son datos asociados a un documento digital que recogen información fundamentalmente descriptiva (autor, título y fecha de creación). También pueden incluir información de administración (creación del recurso, derechos y control de acceso), y preservación (tipo de formato) (Universidad Carlos III de Madrid). Se define actualización de metadatos de media al proceso de actualizar la información descriptiva del medio audio visual, con el fin de mantener la meta-información de la media actualizada con respecto al contenido de la misma.

1.2.8 Técnicas para balanceo de cargas operacionales

Las técnicas para balanceo de cargas operacionales en ambientes paralelos y distribuidos son el desarrollo de técnicas eficientes para distribuir procesos de un programa paralelo sobre múltiples procesadores (Jiménez, 2010). El balance de carga es una técnica que acrecienta los recursos, explotando el paralelismo, acortando el tiempo de respuesta mediante una distribución apropiada de la aplicación (Soto, 2008).
Las técnicas para balanceo de cargas operacionales son diseñadas para la distribución equitativa de tareas a realizar por sistemas u ordenadores, a partir de un algoritmo que tiene en cuenta los distintos atributos que estos poseen, como ejemplo la disponibilidad de procesamiento y almacenamiento.

1.2.9 Persistencia de datos

Para definir persistencia de datos es necesario primero enunciar que son los datos. Para la informática, los datos son expresiones generales que describen características de las entidades sobre las que operan los algoritmos. Estas expresiones deben presentarse de una cierta manera para que puedan ser tratadas por una computadora. En estos casos, los datos por si solo tampoco constituyen información, sino que esta se obtiene al realizar un adecuado procesamiento de los datos (Definición De, 2011). En la presente investigación la persistencia de datos se define como la posibilidad de perdurar que debe poseer un objeto de aplicación en el que están contenidos información referente a los procesos realizados por el sistema, estos procesos pueden estar ejecutándose localmente o en ordenadores interconectados en una red de datos. La persistencia puede ser lograda al serializar en ficheros los objetos o almacenándolos en una base de datos.

1.2.10 Tolerancia a fallos

RAE conceptualiza fallo como falta, deficiencia o error (ESPAÑOLA, 2011). La tolerancia a fallos es la capacidad para mantener los datos disponibles en caso de que se produzcan uno o varios fallos en el sistema (Torrecillas). Existen tres tipos de tolerancia de fallos:

- Tolerancia completa (*fail operational*).
  El sistema sigue funcionando, al menos durante un tiempo, sin perder funcionalidad ni prestaciones.

- Degradación aceptable (*fail-soft*).

El sistema sigue funcionando con una pérdida parcial de funcionalidad o prestaciones hasta la reparación del fallo.

- Parada segura (*failsafe*).
  El sistema se detiene en un estado que asegura la integridad del entorno hasta que se repare el fallo.

El grado de tolerancia de fallos necesario depende de la aplicación (Alvarez, 2011).
La tolerancia a fallos es la habilidad que puede poseer un sistema de seguir funcionando luego de que suceda algún evento inesperado, como pueden ser fallos de hardware, de software o de interrupción de energía eléctrica al ordenador. Esta característica es muy necesaria en sistemas de gran envergadura, pues mientras mayor sea la cantidad de información procesada en un ambiente de riesgo, mayores serán las pérdidas ocurridas ante algún incidente.

1.3 Objeto de Estudio

Se define como **objeto de estudio** los procesos de gestión de materiales audiovisuales.

1.3.1 Descripción General

El Sistema Gestor de Procesos de Media es una aplicación para la integración de sistemas en entornos broadcast[1], que brinda mediante una plataforma de indexación y codificación funcionalidades para realizar varios procesos sobre materiales audio-visuales. Con esta nueva versión del sistema se pretende crear un software capaz de automatizar e integrar los procesos, operaciones, estaciones de trabajos y equipamiento en la medida que sea posible, donde existan entornos colaborativos, descentralizados y en general para cualquier negocio relacionado con la producción y gestión de audio-visuales.

El Sistema Gestor de Procesos de Media v2 estará formado por dos subsistemas, el Gestor de Procesos de Media y la Plataforma de Codificación e Indexación, en esta última estarán disponibles en forma de servicios las distintas operaciones o procesos a realizarse. Los procesos que se realizan sobre las medias en la Plataforma de Codificación e Indexación: son codificación, indexación, transferencia y actualización de metadatos. El Gestor de Procesos de Media publicará un servicio, mediante este, distintas aplicaciones clientes solicitarán la ejecución de flujos de procesos. Estos flujos

[1] Broadcast: En el ámbito de la Televisión. Señala existencia de transmisión de información mediante una red de datos.

son grupos de procesos que podrán ser ejecutados en los nodos esclavos, con una prioridad y un orden de ejecución especificado mediante configuraciones en el Gestor de Procesos de Media. A través de este servicio las aplicaciones clientes también podrán monitorizar los flujos gestionados en el sistema así como los nodos esclavos pertenecientes a la granja de servidores del mismo. Un flujo puede contener unos o varios procesos de distintos tipos. Luego de creado un flujo se asigna la ejecución de los procesos de este a los distintos nodos esclavos, el Gestor de Procesos de Media será el encargado de gestionar todas las tareas que asignen a la Plataforma de Codificación e Indexación, o a los diversos nodos dedicados que podrán estar ejecutándose en uno o varios ordenadores provistos de altas prestaciones de hardware. Para el desarrollo del sistema se empleará una arquitectura orientada a servicios, es necesaria pues se estará en presencia de equipamiento y ordenadores interconectados en una red de datos que necesiten procesar y comunicar información, el sistema deberá ser flexible al mismo tiempo en cuanto a escalabilidad e integración.

Se pretende elaborar una Plataforma de Codificación e Indexación a la que se puedan agregar los distintos tipos de operaciones de gestión de audiovisuales (Ejemplo: transferencia, codificación de audio y video e indexación) en forma de plugins, en los que estarán contenidas las funcionalidades e instrucciones. Este desarrollo basado en plugins posibilitará al sistema realizar otras operaciones de procesamiento de medias, que le serán agregados en cualquier momento una vez funcionando está nueva versión del sistema, facilitando grandes niveles de escalabilidad y una elevada posibilidad de integración.

Las distintas funcionalidades que serán agregadas a la Plataforma de Codificación e Indexación, estarán gestionadas por el Gestor de Procesos de Media. Para lograr esta integración, en un inicio este último no tendrá información referente a ninguno de los servicios publicados en la plataforma, se deberá leer un fichero de configuraciones que proveerá la información necesaria para crear flujos de trabajos compuestos por distintos procesos que pueden ser tanto de procesamiento como de notificación. Los mismos serán organizados de acuerdo a su prioridad y para agregar nuevas funcionalidades al sistema, solo será necesario proveer plugins y ficheros de configuraciones evitando la necesidad de realizar modificaciones al código fuente.

El Gestor de Procesos de Media, en el proceso de gestión de los flujos de procesos que serán definidos, tendrá que realizar un balance de cargas operacionales de forma tal que no se inunde de tareas algún nodo, mientras que otro esté ocioso. De los nodos será necesario conocer su disponibilidad y capacidad de procesamiento

El Gestor de Procesos de Media podrá gestionar nodos de dos tipos, Plataforma de Codificación e Indexación y nodos en los que se estén ejecutando sistemas desarrollado por terceros, estos no poseerán la lógica para interactuar con el Gestor de Procesos de Media. La Plataforma de Codificación e Indexación notificará al Gestor de Procesos de Media la información necesaria para que este pueda realizar su funcionamiento, esta información se notificará a través de un servicio que el Gestor de Procesos de Media publicará. Los nodos donde se estén ejecutando sistemas terceros no poseerán la lógica necesaria para notificar ninguna información al Gestor de Procesos de Media, esta información será proporcionada entonces en forma de ficheros de configuración que el Gestor de Procesos de Media cargará una vez que se haya ejecutado.

En el desarrollo de sistemas informáticos de gran envergadura, existen dos aspectos que son importantes, la persistencia de datos y la tolerancia a fallos. El primero es indispensable tanto en el Gestor de Procesos de Media como en la Plataforma de Codificación e Indexación, dado que si ocurriera un caso siniestro, la información concerniente a los procesos que estaba realizando el sistema estará siempre disponible. La tolerancia a fallos es necesaria pues si un nodo está realizando un flujo de actividades y sufre algún error de software o de hardware que impida la continuidad de su correcto funcionamiento, el Gestor de Procesos de Media deberá ser capaz de detectar este error y reasignar la operación a otro nodo, así mismo en caso de avería debe poseer la capacidad de una vez reiniciado continuar las tareas que tenía pendientes. Por su parte la Plataforma de Codificación e Indexación debe ser capaz de detectar problemas de desconexión del Gestor de Procesos de Media, sin que se afecten los posibles procesos que se podrán estar ejecutando en la Plataforma de Codificación e Indexación y a su vez esta tiene que ser capaz de mantener un secuencia dinámica de peticiones al Gestor de Procesos de Media para poder detectar la conexión del mismo.

1.3.2 Descripción actual del dominio del problema

Actualmente en el departamento de Señales Digitales se encuentra desarrollada la primera versión del Sistema Gestor de Procesos de Media, solución para el procesamiento distribuido de archivos multimedia. Existe un subsistema nombrado Plataforma de Codificación e Indexación que realiza tareas de procesamiento de medias, el mismo puede estar instalado en uno o varios servidores distribuidos geográficamente. A la Plataforma de Codificación e Indexación llegan peticiones provenientes del Gestor de Procesos de Media a través de servicios publicados, junto a estas peticiones llegan los parámetros necesarios para ejecutar los componentes específicos (plugins) en la Plataforma de Codificación e Indexación. A medida que se van ejecutando las tareas en la Plataforma

de Codificación e Indexación, esta mantendrá actualizando al Gestor de Procesos de Media, a través de peticiones a sus servicios publicados, con la evolución de las tareas asignadas por el mismo; en caso de detectarse problemas con el Gestor de Procesos de Media, la Plataforma de Codificación e Indexación continuará con la ejecución de los procesos hasta que estos culminen.

Por su parte al Gestor de Procesos de Media llegan peticiones provenientes de usuarios externos (sistemas informáticos) a través de sus servicios publicados, este los procesa y asigna tareas a distintos servidores donde se ejecuta la Plataforma de Codificación e Indexación; en la medida que se vaya ejecutando cada tarea, se da paso al inicio de otras. La asignación de tareas se realiza de forma equilibrada, teniendo en cuenta la capacidad de procesamiento y la carga de trabajo de los servidores que estén funcionando en el sistema, evitando la sobrecarga para unos y un estado ocioso en otros. Si un servidor donde esté ejecutándose la Plataforma de Codificación e Indexación está procesando una tarea y por alguna razón deja de funcionar, el Gestor de Procesos de Media reasigna el proceso interrumpido a otro servidor, garantizando la realización del mismo.

1.3.3 Situación Problemática

La solución existente en el departamento de Señales Digitales para la gestión de procesos de media presenta deficiencias en su arquitectura, solo publica servicios para aplicaciones que se comunican mediante objetos remotos que empleen ICE como tecnología de comunicación. Esto obliga a aplicaciones externas a usar esta tecnología (ICE), constituyendo un inconveniente pues esta tecnología no es un estándar de comunicación, además obliga a las soluciones que la empleen en su funcionamiento a seguir la política de Licencia Pública General o GPL como también es conocida por sus siglas en inglés, esta obliga a entregar el código fuente del software para su libre modificación y distribución a terceros; para liberarse de esta atadura sería necesaria la adquisición de la licencias comerciales costosas.

En su funcionamiento el sistema Gestor de Proceso de Media luego de recibir una petición, crea un flujo de trabajo, que no es más que varios procesos a ejecutar en un orden y con prioridad definida por el programador. Esto le quita flexibilidad a este sistema ya que para agregar una nueva operación al flujo o cambiar el orden de ejecución de las mismas habría que modificar gran cantidad de código. Se perdería tiempo en que un programador se estudie el funcionamiento de un desarrollo creado por un tercero. Otra deficiencia del sistema está en la forma de realizar el almacenamiento de los datos; una base de datos SQLite es quien almacena de manera temporal el estado de los procesos y flujos que el gestor posee en ejecución. Además se guardan los datos de los nodos esclavos (tipo Plataforma de

Codificación e Indexación). Esta caché física no permite la inclusión de nuevas tablas en caso de que se quiera agregar otro tipo de operación o nodo. Es necesaria una estructura de bases de datos dinámica que soporte estos cambios y sea flexible.

El Gestor de Proceso de Media está limitado a manipular solo nodos esclavos del tipo plataforma de codificación. Cuando se necesita añadir otro nodo como por ejemplo un grabador de video, el Gestor de Proceso de Media no es capaz de soportarlo pues su lógica no lo admite. Además no permite tampoco tener en su gestión nodos que no sean computadoras, ejemplo una cámara de video. El Gestor de Proceso de Media utiliza una estrategia de descubrimiento en la red. Consiste en un servicio que es capaz de recibir al nodo esclavo que desee formar parte de la granja de nodos del Gestor de Procesos de Media. En este caso: ¿cómo una cámara grabadora de video va a formar parte de esta granja si no tiene una lógica que permita esto? Esta filosofía del gestor debe cambiar para que pueda interactuar con equipamiento de hardware y software producidos por terceros.

Por su parte la Plataforma de Codificación e Indexación está limitada a soportar lógicas de análisis, indexación y codificación de materiales audio-visuales, en casos de ser necesaria la inclusión de nuevas lógicas al sistema, esto conllevaría a grandes cambios en el código fuente de la aplicación. Otra deficiencia que presenta este sistema, está marcada por la incapacidad de comunicación entre la Plataforma de Codificación e Indexación y el Gestor de Procesos de Media con el uso de tecnologías distintas a ICE.

Estas particularidades convierten al Sistema Gestor de Proceso de Media en un débil competidor en el mercado actual pues no posee una capacidad rápida de adaptación. Al surgir nuevos clientes con distintas necesidades no es posible de manera rápida satisfacer las mismas, por lo que seguramente el cliente ira por otras opciones existentes, realizar una personalización constituye un proceso engorroso que consumiría tiempo, una espera por el desarrollo de un sistema se traduce en pérdidas monetarias para las partes interesadas, además los desarrollos implican gasto de recursos y personal calificado, que pudiera destinarse a otras actividades.

1.4 Análisis de otras soluciones existentes

1.4.1 TDIndexer

TDIndexer (Módulo de indexación y catalogación automática de Tedial) dispone de funcionalidades de indexación de storyboard, de thesaurus, de búsqueda avanzada, de conversor de audio a texto, de

assets virtuales[2] y reconocimiento de patrones. Las funcionalidades conforman un conjunto de programas que analizan los archivos multimedia y extraen de manera automática información fundamental de acceso para catalogación y edición. El acceso directo a la información audio visual digital requiere el desarrollo de herramientas eficientes para la indexación y la catalogación automática de los contenidos multimedia. Esto redundará en una reducción drástica de los costes de explotación de los sistemas digitales de información audio visual.

Características Principales

- Indexación de storyboard y thesaurus.
- Conversor de audio a texto.
- Búsqueda avanzada, assets virtuales y reconocimiento de patrones.
- Diversidad de formatos, catalogación y acceso inmediato a cualquier fragmento del archivo:

TDIndexer no sólo realiza tareas rutinarias de forma automática, sino que además ofrece al usuario un abanico de herramientas avanzadas que facilitan la gestión posterior de la información, incluyendo los módulos siguientes:

- **Módulo de indexación:** Analiza el contenido del archivo y extrae información del formato de compresión, de la imagen, del audio y/o el texto. Además, genera un conjunto auxiliar de archivos que permiten la manipulación completa del contenido multimedia. TDIndexer posee módulos de indexación para los formatos multimedia siguientes: MPEG (-1,-2 y 4), DV (-25y50), GXF, ASF, AVI, WAV, PDF y RTF.

- **Conversor:** La transcripción de una señal de audio a un fichero de texto puede utilizarse como fuente de metadatos para la catalogación automática de material audio visual. Incluye un módulo de conversión desarrollado a partir de la herramienta Via Voice que permite convertir audio a texto independientemente del locutor.

- **Thesaurus:** La catalogación avanzada de los contenidos multimedia requiere el uso de bases de datos de conocimiento especializadas. TDIndexer incluye un módulo thesaurus que permite al cliente crear bases de datos de conocimiento para cada uno de sus dominios de interés, lo que permite localizar dichos contenidos posteriormente de forma eficiente mediante una búsqueda avanzada (TEDIAL Tecnologías Digitales Audiovisuales, S.L.).

[2] Assets virtuales: permite agrupar fragmentos de distintos archivos en una única unidad de contenido (asset virtual) sin replicar el material multimedia.

1.4.2 TDMPM

TDMPM es un gestor de procesos de media, el principal componente de la solución de Tedial para el tapeless[3]. Define, planifica y monitoriza los flujos de media entre los servidores distribuidos y sistemas. Los formatos de almacenamiento se optimizan para satisfacer del mejor modo posible las necesidades de la organización. Esto quiere decir que todas las transferencias de media y su manejo se efectúan como datos, a la vez que el número de procesos de codificación, transcodificación y mutación del encapsulado se minimizan para mantener un alto rendimiento sin que la calidad se vea comprometida. El resultado es un sistema de flujos de contenido integrado y automatizado en el que cualquier intercambio de media y metadatos se realiza a través de la red.

MPM es un sistema de software que optimiza y automatiza cualquier transferencia de media. Su integración de interfaz probada con equipos de terceros permite construir sistemas homogéneos e integrados usando la mayoría de equipos existentes y manteniendo lo mejor de los procesos de trabajo del cliente. Los datos se transfieren a través de una red de datos que incluye el encapsulado y desencapsulado siempre que sea necesario sin perder ninguna información durante el proceso. Cuando el intercambio de datos requiere de una nueva codificación externa, MPM automatiza esta transferencia de archivo de transcodificación y el control remoto del dispositivo externo; además, de encargarse de la actualización de las bases de datos y preservar cualquier metadato asociado a la media.

Características principales de MPM

- **Integración:** MPM puede integrarse con cualquier dispositivo de terceros a través de su API específico.
- **Escalabilidad:** La capacidad del MPM puede verse incrementada simplemente añadiendo nuevos servidores.
- **Configuración:** Sencilla configuración de usuario gracias al Flow Builder Editor que permite realizar cualquier cambio en cualquier momento de forma temporal o permanente.
- **Estructura del formato y restauración:** Verificación del formato estándar y restauración siempre que sea necesaria.
- **Pre catalogación:** MPM asocia los metadatos a la media durante el proceso de ingesta.
- **Sincronización de la base de datos:** MPM intercambia metadatos con bases de datos externas.

[3] Sistema tapeless: En televisoras. Un sistema basado en entornos de almacenamiento sin cintas de vídeo.

- **Web:** MPM introduce la publicación Web (TEDIAL Tecnologías Digitales Audiovisuales).

1.4.3 Pro Media Carbon

ProMedia Carbon se basa en archivos de software de transcodificación que facilita la conversión de los medios de comunicación a la mayor variedad de adquisición, edición, difusión, Web y formatos móviles disponibles. Las escalas del sistema para ajustar con precisión los requisitos iníciales y crece a un sistema automatizado de soporte multi-nodo de la comunidad de transcodificación bajo el control de Carbon Server o el sistema de flujo de trabajo (CMA). ProMedia Carbon también incluye una API abierta que permite la creación de flujos de trabajo personalizados o aplicaciones de tercera parte.

Como parte del proceso de transcodificación, ProMedia Carbon maneja una amplia gama de operaciones críticas incluyendo SD / HD de conversión PAL / NTSC, inserción de logos, conversión del espacio de color, corrección de color, la extracción de Close Caption y mucho más. ProMedia Carbon proporciona transcodificación de alto rendimiento, escalable y rentable para una amplia gama de ambientes de negocios, desde estudios especializados a nivel de empresa, sus principales características son:

- **Basado en archivos de software de transcodificación:** Utiliza un motor de flujo de trabajo basado en ficheros. Todas sus habilidades se basan en la adquisición, transformación y entrega de archivos.

- **Amplia compatibilidad de formatos:** Incluido a ProMedia Carbon está la capacidad de codificar a partir de prácticamente cualquier formato de producción de los medios de comunicación a todos los estándares de los medios de comunicación en la actualidad.

- **Interfaz de usuario intuitiva:** La Interfaz de ProMedia Carbon da un control completo sobre todos los aspectos del proceso de transcodificación.

Operación automatizada.

ProMedia Carbon se puede ejecutar en un modo totalmente automático con soporte para procesamiento por lotes, carpetas de vigilancia, y automática de las transferencias FTP. También ofrece transcodificación inteligente para aumentar la productividad al permitir la identificación de formatos de origen y de forma inteligente a la transcodificación de formatos de destino deseado de forma automática.

- **La transcodificación escalables:** Para grandes tareas de transcodificación, varios software ProMedia Carbon se pueden configurar como una granja de transcodificación, bajo el control del software Carbon Server.

- **Basado en XML SDK:** ProMedia Carbon puede ser controlado directamente a través de un SDK basado en XML proporcionado con el software. Todos los aspectos del proceso de transcodificación puede ser controlado por el SDK, incluyendo origen y destino, los parámetros de transcodificación, filtrado, composición, inserción de anuncios, la titulación y las notificaciones (Harmonic Inc., 2011).

1.5 Conclusiones Parciales

Después de haber abordado los principales conceptos del dominio del problema actual, se logra tener una idea más aterrizada a la investigación que se propone. Observando las principales características de sistemas similares como MPM, se identifica que la solución de software a desarrollar en la presente investigación debe permitir la integración de sistemas con sistemas y sistemas con equipamiento de hardware. Por tanto la solución a desarrollar debe ser integrable con ambientes ya existentes en distintos negocios y brindando la posibilidad de satisfacer las necesidades que puedan tener los diversos clientes.

CAPÍTULO 2: Herramientas y Tecnologías a utilizar

2.1 Introducción

En el presente capítulo se definen tecnologías actuales, que son empleadas en el desarrollo de software a nivel mundial. Se argumentan además las metodologías, arquitecturas y herramientas utilizadas en la elaboración de la solución que conlleva a la presente investigación.

2.2 Metodología de desarrollo de software

Una metodología de desarrollo de software es un conjunto de pasos y procedimientos que deben seguirse para desarrollar software. Una metodología está compuesta por:

- Cómo dividir un proyecto en etapas.
- Qué tareas se llevan a cabo en cada etapa.
- Qué restricciones deben aplicarse.
- Qué técnicas y herramientas se emplean.
- Cómo se controla y gestiona un proyecto (Hernando, 2001)**.**

En el ámbito de las metodologías de software, se encuentran dos clasificaciones, las nombradas metodologías pesadas, que están destinadas para el control de los procesos, especificando con inflexibilidad todas las actividades a realizar, y expresa además las herramientas que se utilizarán, y las metodologías ligeras, orientadas a la interacción del cliente y la evolución incremental del producto. En cortos intervalos de tiempo los clientes pueden ir apreciando las versiones del software que se van desarrollando, con el objetivo de poder realizar cambios según las necesidades del cliente. Dentro de estas clasificaciones de metodología se pueden mencionar como ejemplos al Proceso Unificado de Desarrollo (RUP), como metodología pesada y Extreme Programming (XP), CRYSTAL y SCRUM como metodología ágiles.

2.2.1 RUP como metodología a utilizar

Proceso Unificado de Desarrollo (*Rational Unified Process* en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización (Phylum, 2012).

Sommerville define RUP como un modelo de proceso de software genérico, que presenta los desarrollos de software como una actividad iterativa de cuatro fases, en que las fases son: creación (Inicio), elaboración, construcción y transición. Creación establece los casos de uso del negocio para el sistema, elaboración define la arquitectura, la construcción implementa el sistema y la transición despliega el sistema en el entorno del cliente (Sommerville, 2006).

Se puede hacer mención de las tres características esenciales que definen al RUP:

Proceso Dirigido por los Casos de Uso: Con esto se refiere a la utilización de los Casos de Uso para el desenvolvimiento y desarrollo de las disciplinas con los artefactos, roles y actividades necesarias. Los Casos de Uso son la base para la implementación de las fases y disciplinas del RUP. Un Caso de Uso es una secuencia de pasos a seguir para la realización de un fin o propósito, y se relaciona directamente con los requerimientos, ya que un Caso de Uso es la secuencia de pasos que conlleva la realización e implementación de un Requerimiento planteado por el cliente.

Proceso Iterativo e Incremental: Es el modelo utilizado por RUP para el desarrollo de un proyecto de software. Este modelo plantea la implementación del proyecto a realizar en Iteraciones, con lo cual se pueden definir objetivos por cumplir en cada iteración y así poder ir completando todo el proyecto iteración por iteración, con lo cual se tienen varias ventajas, entre ellas se puede mencionar la de tener pequeños avances del proyectos que son entregables al cliente el cual puede probar mientras se está desarrollando otra iteración del proyecto, con lo cual el proyecto va creciendo hasta completarlo en su totalidad.

Proceso Centrado en la Arquitectura: Define la arquitectura de un sistema, y una arquitectura ejecutable construida como un prototipo evolutivo. La arquitectura de un sistema es la organización o estructura de sus partes más relevantes. Una arquitectura ejecutable es una implementación parcial del sistema, construida para demostrar algunas funciones y propiedades. RUP establece refinamientos sucesivos de una arquitectura ejecutable, construida como un prototipo evolutivo (CHACÓN, 2003).

En la Figura 1 se observan las distintas proporciones de trabajo en cada disciplina de RUP, en correspondencia con las fases e iteraciones.

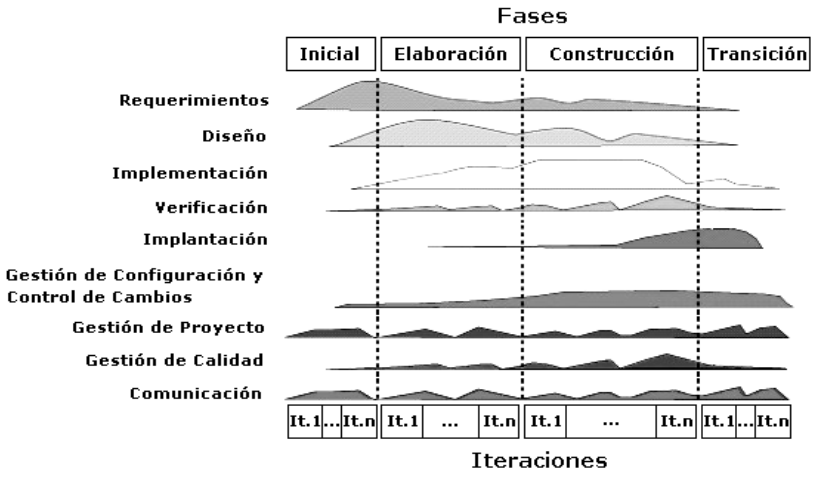

Figura 1 Disciplinas, Fases e Iteraciones de RUP) *(Vallespir, 2011)*.

RUP proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización del desarrollo. Su meta es asegurar la producción del software de alta calidad que resuelve las necesidades de los usuarios dentro de un presupuesto y tiempo establecidos.

2.3 UML

Hoy en día, el Lenguaje Unificado de Modelado (UML del inglés *Unified Modeling Language*) está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código. En otros términos, así como en la construcción de un edificio se realizan planos previo a su construcción, para el desarrollo de Software se deben realizar diseños en UML previamente a la codificación de un sistema, ahora bien, aunque UML es un lenguaje, este posee más características visuales que programáticas, estas facilitan a integrantes de un equipo multidisciplinario participar e intercomunicarse fácilmente (Osmosislatina.com, 2007).

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otros desarrolladores lo puedan entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema

desarrollado que pueden servir para su futura revisión (Orallo, 2012).

Se empleará UML por las bondades que brinda a los desarrolladores, al proveer un vocabulario y reglas para permitir una comunicación estándar, para visualizar, construir y documentar todo lo referente al proceso de desarrollo, añadiendo a esta última robustez, aumentando las posibilidades de obtener un producto exitoso.

2.4 Visual Paradigm como herramienta CASE

Se considera una Herramienta CASE a la herramienta o sistema informático que emplean los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software. El uso de Herramientas CASE en el desarrollo de un proyecto, garantiza un aumento en la calidad de los desarrollos realizados y por consiguiente, un aumento de la productividad. Ejemplo de este tipo de herramientas son: *System Architect, Rational Rose y Visual Paradigm.*

Para el modelado de los diagramas se empleará la herramienta Visual Paradigm, está diseñada para ayudar al desarrollo de software y soporta los principales estándares de la industria tales como UML.

Es una potente herramienta de modelado, lo que le hizo meritoria del premio Jolt 2011 para el Diseño, la Arquitectura, y herramientas de planificación. Ofrece un conjunto completo de herramientas a los equipos de desarrollo de software, necesario para la captura de requisitos, software de planificación, la planificación de controles, la clase de modelado, modelado de datos (Visual Paradigm, 2012).

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre de probada utilidad para el analista. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (EcuRed, 2012).

Visual Paradigm provee además de un conjunto de tutoriales y ayudas para la creación de sistemas informáticos, es una herramienta fácil de utilizar. Contiene una conexión con la herramienta Rational Rose en sus archivos de proyecto (.MDL / .CAT) los cuales pueden ser importados a Visual Paradigm UML. Otras características significativas que hacen que se opte por esta herramienta son:

- Software libre.
- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- Licencia: gratuita y comercial.
- Modelado colaborativo con CVS y Subversión (control de versiones).
- Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de código - Modelo a código, diagrama a código.
- Diagramas de flujo de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación (Visual Paradigm, 2012).

2.5 Lenguaje de programación a utilizar

Un **lenguaje de programación** es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación (Norman, 2011).

Existen diferentes lenguajes de programación, cada uno posee sus características específicas. Entre los más sobresalientes se conocen algunos como C, C++, C#, Java, Java Script, ASP y PHP.

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multi paradigma (Brokken, 2012).

2.6 Entorno Integrado de Desarrollo

Un **Entorno de Desarrollo Integrado** (del inglés *Integrated Development Environment* o IDE) es un programa compuesto por una serie de herramientas que utilizan los desarrolladores para escribir el código. Puede soportar varios lenguajes de programación o puede estar diseñada para uno únicamente. Existen muchos ejemplos de IDE lo constituyen: NetBeans, Delphi de Borland, MS Visual Studio .NET de Microsoft y Qt Creator.

2.6.1 Qt Creator como IDE de desarrollo

Qt Creator es un IDE multi-plataforma, se ejecuta en Windows, Linux/X11 y Mac OS, sistemas operativos de escritorio, y permite a los desarrolladores crear aplicaciones tanto para escritorio, como para plataformas de dispositivos móviles.

Principales características de Qt Creator:

- Editor de código sofisticado: Avanzado editor de código de Qt Creator ofrece soporte para la edición de C + + y QML (Java Script), ayuda sensible al contexto, completado de código y navegación.
- Control de versiones: Qt Creator se integra con la mayoría de los sistemas de control de versiones populares, incluyendo Git, Subversion, Bazaar, Perforce, CVS y Mercurial.
- Los diseñadores de interfaz de usuario integrada: Qt Creator proporciona dos editores integrados visuales: Qt Designer para la creación de interfaces de usuario de widgets Qt y Qt Designer para el desarrollo de interfaces de usuario rápido de animación con el lenguaje QML.
- Construcción y gestión proyecto: Si se importa un proyecto existente o crear uno desde cero, Qt Creator genera todos los archivos necesarios. Apoyo a CMake y compilación cruzada, con qmake está incluido.
- Aplicaciones de escritorio y móvil: Qt Creator ofrece soporte para crear y ejecutar aplicaciones Qt para equipos de escritorio y dispositivos móviles. Las opciones de construcción permiten cambiar rápidamente los destinos de generación.

- Simulador de Qt: Disponible como parte del SDK de Qt, el Simulador de Qt permite probar la aplicación para dispositivos móviles en un entorno similar a la del dispositivo de destino (Nokia Corporation, 2008).

2.7 Sistema gestor de base de datos

Un sistema gestor de base de datos (SGBD) es una aplicación que permite al usuario realizar un conjunto de acciones (definir, crear y administrar) vinculadas con el manejo de los datos contenidos en una base de datos. Con su uso se puede proveer privacidad, integridad y seguridad a los datos. Una característica distintiva es que emplean un lenguaje basado en consultas. Algunos ejemplos de SGBD: MySQL, Oracle, MS SQL Server, PostgreSQL y SQLite.

2.7.1 PostgreSQL y SQlite como sistemas gestores de base de datos

Se inicia en la Universidad de Berkeley en 1977 bajo el nombre Ingres, en el año 1986, cambia su nombre a Postgres con el objetivo de aplicar los conceptos de Objetos Relacionales y no es hasta 1995, que cambia su nombre a Postgres95 que luego derivaría a PostgreSQL.
Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD lo que permite al contrario de GPL, que se puede emplear su código fuente en software no libre (PostgreSQL Global Development Group , 2012).

Características de PostgreSQL

- Atomicidad (Indivisible) es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- Consistencia es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- Aislamiento es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error.
- Durabilidad es la propiedad que asegura que una vez realizada la operación, esta persistirá y no se podrá deshacer aunque falle el sistema.
- Corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos y Windows.

- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.
- Comunidades muy activas, varias comunidades en castellano.
- Altamente adaptable a las necesidades del cliente.
- Soporte nativo para los lenguajes más populares del medio: ejemplo PHP, C, C++, Perl y Python.
- Soporte de todas las características de una base de datos profesional (Quiñones, 2011).

2.7.2 SQLite

SQLite es una librería de software que implementa auto contenido, sin servidor, cero configuraciones y motor de base de datos SQL transaccional. SQLite es el más utilizado motor de base de datos SQL en el mundo. El código fuente de SQLite es de dominio público. El desarrollo y mantenimiento de SQLite es patrocinado en parte por miembros del Consorcio SQLite, entre ellos se puede encontrar Mozilla, Adobe, Oracle y Nokia.

Es un sistema integrado de base de datos SQL. A diferencia de la mayoría de las otras bases de datos SQL, SQLite no tiene un proceso servidor independiente. SQLite lee y escribe directamente a los archivos de disco normal. Una completa base de datos SQL con varias tablas, índices, triggers y vistas, está contenida en un archivo de disco único. El formato de archivo de base de datos es multi-plataforma, por lo que libremente se puede copiar una base de datos entre los sistemas de 32-bit y 64-bit o entre big-endian y little-endian arquitecturas. Estas características hacen de SQLite una elección común como un formato de archivo de aplicación (SQLlite.org, 2012).

Luego de conocer y evaluar las principales funcionalidades que ofrecen estos dos gestores de bases de datos, para lograr el desarrollo eficiente del sistema, se selecciona SQLite como sistema gestor de base de datos, debido a que es un sistema multiplataforma, con un alto nivel de portabilidad al no depender de ningún tipo de servidor, el volumen de información que será necesario almacenar será de poca envergadura, y posee alta velocidad de respuesta considerándose hoy en día unos de los sistemas gestores de base de datos más rápidos del mundo.

2.8 XMLRPC como tecnología de comunicación

XML–RPC es un protocolo de llamada remota a procedimientos que funciona sobre Internet, mediante el intercambio de mensajes entre cliente y servidor, utilizando el protocolo HTTP para el transporte de los mensajes. XML–RPC utiliza peticiones POST de HTTP para enviar un mensaje, en formato XML,

señalando el procedimiento que se va a ejecutar en el servidor, los parámetros; y posteriormente devuelto por el servidor el resultado en formato XML (XMLRPC.COM, 1999).

XML-RPC constituye además un lenguaje de mensajería basada en XML, estandarizados por el consorcio W3C, que especifican todas las reglas necesarias para ubicar servicios Web XML, integrarlos en aplicaciones y establecer la comunicación entre ellos. XML-RPC está diseñado para ser sencillo, la curva de aprendizaje de XML-RPC es muy suave, por lo que un programador novato en este campo, puede conseguir resultados satisfactorios con poco esfuerzo (WEBNOVA, 2011).

Debido a la problemática actual con las licencias de software del middleware ICE, el cual posee dos tipos de licencias una Comercial, altamente costosa y otra libre licenciada bajo GPL (*General Public Licence*) en su versión 3.0, la cual obliga a los desarrolladores que la utilizan, a entregar el código fuente al cliente y a publicarlo a toda la comunidad.

Debido a esta situación surge la necesidad de utilizar herramientas factibles, que garanticen el desarrollo eficiente del sistema, eligiéndose como opción viable la utilización del estándar de comunicación XMLRPC, garantizando compatibilidad y alta capacidad de integración. En cuestiones de licencia no existen limitantes, pues esta tecnología es totalmente libre, además ofrece la ventaja de estar desarrollado en una biblioteca llamada *libqxmlrpc* para que aplicaciones desarrolladas en el framework Qt puedan utilizar este protocolo estandarizado desde hace décadas, integrado al IDE seleccionado para el desarrollo del sistema.

2.9 BPEL

BPEL, siglas en inglés de (*Business Process Execution Language*), es un lenguaje de programación destinado para la ejecución de Procesos Empresariales. BPEL es un descendiente de WSFL y XLANG, y se deriva de XML. También es conocido como WS-BPEL o Web Services Business Process Execution Language (Lenguaje de Ejecución de Procesos de Negocio con Servicios Web). El lenguaje fue concebido por grandes de la informática como Oracle, BEA Systems, IBM, SAP, Microsoft, y estandarizado por OASIS (Organización para el Avance de Estándares de Información Estructurada). Es un lenguaje de alto nivel que lleva el concepto de servicio un paso adelante al proporcionar métodos de definición y soporte para flujos de trabajo y procesos de negocio (OASIS, 2007).

2.10 Conclusiones Parciales

Una vez conocidas las principales características de las tecnologías y herramientas a emplear, se puede afirmar que poseen los elementos para satisfacer las necesidades existentes y que el uso de las mismas puede aportar fortaleza a la concepción de la solución, brindando organización, flexibilidad y control durante todo el desarrollo de software. Al emplear herramientas libres, se logrará independizar la solución de licencias propietarias costosas.

CAPÍTULO 3: Descripción de la solución propuesta

3.1 Introducción

Este capítulo se centra en describir conceptos relacionados con las definiciones de estilo y patrones de diseño de software, estilos de programación, estándar de codificación empleado. Se caracteriza la solución propuesta y se valora el diseño de clases propuesto por el analista.

3.2 Patrones y Estilos de Diseño de Software

Los patrones de software no son más que un conjunto de soluciones a problemas habituales en el diseño de software.

“Un patrón es una solución de diseño de software a un problema, aceptada como correcta, a la que se ha dado un nombre y que puede ser aplicada en otros contextos”. Un patrón captura la experiencia y conocimiento de expertos, quienes han producido soluciones exitosas a problemas, a fin que esas soluciones queden a disposición de personas con menos experiencia; sin embargo, los patrones no proveen siempre las soluciones definitivas, algunas veces, los usuarios de patrones deben tener creatividad para utilizar, instanciar o implementar un patrón (Bonillo, 2006).

3.2.1 Patrones de Diseño GOF empleados

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Se debe tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

Existen tres tipos fundamentales de patrones:

- Patrones Creacionales: Inicialización y configuración de objetos.
- Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos (Tedeschi, 2012).

3.2.1.1 Patrones Creacionales

Fábrica abstracta *(Abstract Factory):* Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí, haciendo transparente el tipo de familia concreta que se esté usando (Torres, 2008).

El uso de este patrón satisface la necesidad de representar a nivel de objetos, las de distintos flujos de operaciones que serán gestionados en el Sistema Gestor de Procesos de Media v2.

Instancia única (*Singleton*) es un patrón que permite tener una única instancia de un objeto en toda la aplicación. El patrón Singleton garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a esta instancia. Da solución al problema de que varios clientes distintos, precisan referenciar a un mismo elemento y se pretende asegurar de que no hay más de una instancia de ese elemento.

El funcionamiento de este patrón es muy sencillo y podría reducirse a los siguientes conceptos:

- Ocultar el constructor de la clase Singleton, para que los clientes no puedan crear instancias.
- Declarar en la clase Singleton una variable miembro privada que contenga la referencia a la instancia única que se desee gestionar.
- Proveer en la clase Singleton una función o propiedad que brinde acceso a la única instancia gestionada por el Singleton. Los clientes acceden a la instancia a través de esta función o propiedad (Welicki, 2011).

Se hará uso de este patrón para darle solución a la necesidad de que deba existir exactamente una instancia de una clase y esta deba ser accesible desde cualquier punto de acceso dentro de la lógica.

3.2.1.2 Patrones Estructurales

Objeto Compuesto (*Composite*): este patrón indica cómo se organizan los objetos en memoria para obtener una composición recursiva o lo que es lo mismo una jerarquía en forma de árbol. Su propósito es construir una jerarquía compuesta por dos tipos de objetos: primitivos (nodos hoja) y compuestos (nodos internos). Los objetos compuestos permiten componer objetos primitivos y otros objetos compuestos en estructuras arbitrariamente complejas. A este tipo de jerarquías también se le suele llamar composiciones recursivas y jerarquías parte-todo. También permite a los clientes de los objetos

composición tratar de forma uniforme a los objetos primitivos y a las composiciones, es decir trata de igual forma a los nodos hoja como a los árboles o nodos internos.

Este patrón surge como motivación de los creadores de editores gráficos en dónde es posible construir objetos complejos a partir de otros más simples. El cliente que usa esta jerarquía de objetos complejos (en este caso el Editor) está interesado en tratar de forma uniforme a los objetos primitivos y a los compuestos para simplificar su código (Lado, 2010).

Sera empleado para construir objetos complejos a partir de otros más simples, utilizando para ello la composición recursiva y una estructura de árbol.

3.2.1.3 Patrones de Comportamiento

Observador (Observer): incluido en el grupo de los Patrones de Comportamiento, es empleado en el desarrollo de sistemas, en los que se cuenten con una colección de objetos relacionados y se necesite mantener la consistencia entre estos objetos relacionados, para ello define 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 es proporcionar a los componentes una forma flexible de enviar mensajes de difusión a los receptores interesados.

De esta forma, un subscriptor (también denominado en algunas ocasiones como observador) es el encargado de subscribirse a un publicador. El publicador es a su vez el encargado de notificar que ha sucedido “algo” que requiere la atención de sus subscritores. En este punto, el publicador estará relacionado con todos sus subscriptores, y cuando ocurra “algo”, todos los subscriptores se darán por enterados. Lógicamente, debe haber un mecanismo que permita agregar un subscriptor y eliminar al subscriptor, tanto para que reciba notificaciones como para que deje de recibirlas (Hernandez, 2012).

Este patrón será empleado en el desarrollo de la solución propuesta, pues se ha identificado que existirán partes de la solución donde será necesario mantener actualizada una información en un conjunto de objetos.

3.2.2 Patrones Generales de Software para Asignar Responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a clases, expresados en forma de patrones. A continuación se describen los utilizados en la investigación:

3.2.2.1 Experto (*Expert*)

En el libro UML y Patrones, Craig Larman su autor, describe al patrón Experto como el principio de que una clase debe poseer la información necesaria para cumplir con sus responsabilidades. Consiste en asignarle las responsabilidades a una clase que dispone de la información necesaria para llevar a cabo una tarea, el cumplimiento de una responsabilidad puede requerir de la información distribuida en otras clases de objetos, lo que significa que pueden existir expertos "parciales" que colaboran en la realización de la tarea.
Este patrón permite conservar el encapsulamiento, pues los objetos se valen de su propia información para hacer lo que se les pide. Soporta un bajo acoplamiento, que contribuye a tener un sistema robusto y de fácil mantenimiento.

3.2.2.2 Creador (*Creator*)

Plantea asignarle a una clase la responsabilidad de crear una instancia de otra siempre que se cumpla que una clase B contenga a una clase A, que exista una agregación entre ellas, que la clase B inicialice los datos de la clase A, o que la clase B registra la clase A.

El patrón Creador indica que la clase incluyente del contenedor o registro es idónea para asumir la responsabilidad de crear objetos de la clase contenida. Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. La intención principal de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. Ofrece soporte de bajo acoplamiento, lo que provee mayor reutilización.

3.2.2.3 Bajo Acoplamiento (*Low Coupling*)

Plantea asignar las responsabilidades de modo que se mantenga bajo acoplamiento. Soluciona el problema siguiente: ¿Cómo dar soporte a poca dependencia y a una mayor reutilización?

Craig Larman describe el acoplamiento como la medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras. Un bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, son más reutilizables, acrecientan la oportunidad de una mayor productividad, no son afectadas al realizar cambios en otros componentes y son fáciles de entender por separadas.

3.2.2.4 Alta Cohesión (*High Cohesion*)

Plantea asignar las responsabilidades de modo que se mantenga una alta cohesión. Soluciona el problema siguiente: ¿Cómo mantener controlable la complejidad?

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza una clase con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.
El patrón alta cohesión define que una clase tiene responsabilidades moderadas en un área funcional moderada con las otras para llevar a cabo las tareas. Las clases con una baja cohesión son difíciles de comprender, reutilizar, conservar y son afectadas constantemente por los cambios que se puedan realizar, las que poseen una baja cohesión representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otras clases. Por lo que al utilizar el patrón Alta Cohesión se logra un diseño con mayor claridad, mejoras en las funcionalidades y una mayor reutilización (Larman, 1999).
Los patrones no deben tratarse de forma independiente pues poseen relaciones que mantienen el equilibrio entre clases, el Bajo Acoplamiento y la Alta Cohesión son importantes patrones que equilibran las responsabilidades entre clases y que garantizan que las clases sean creadas con un buen diseño donde los objetos sean capaz de interactuar.

3.3 Patrones Arquitectónicos

Es necesario saber cuándo emplear el término Patrón Arquitectónico o Estilos Arquitectónicos. Los estilos sólo se manifiestan en arquitectura teórica descriptiva de alto nivel de abstracción; los patrones, por todas partes. Los partidarios de los estilos se definen desde el inicio como arquitectos; los que se agrupan en torno de los patrones se confunden a veces con ingenieros y diseñadores, cuando no con programadores con conciencia sistemática o lo que alguna vez se llamó analistas de software (Reynoso, y otros, 2005).

3.3.1 Arquitectura Pizarra o Repositorio

En esta arquitectura hay dos componentes principales: una estructura de datos que representa el estado actual y una colección de componentes independientes que operan sobre él. En base a esta distinción se han definidos dos subcategorías principales del estilo: Si los tipos de transacciones en el flujo de entrada definen los procesos a ejecutar, el repositorio puede ser una base de datos tradicional

(implícitamente no cliente-servidor). Si el estado actual de la estructura de datos dispara los procesos a ejecutar, el repositorio es lo que se llama una pizarra pura o un tablero de control (Bonillo, 2006).

En el diseño del Sistema Gestor de Procesos de Media v2 se emplea la segunda subcategoría, se identifica que esta es la factible para dar solución a la situación existente, y vale resaltar que emplear la colección de componentes independientes que caracteriza a esta arquitectura, provee flexibilidad y escalabilidad.

3.3.2 Arquitectura en Capas

En Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior.

En un estilo en capas, los conectores se definen mediante los protocolos que determinan las formas de la interacción. Las ventajas del estilo en capas son obvias. Primero que nada, el estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. En segundo lugar, el estilo admite naturalmente optimizaciones y refinamientos. En tercer lugar, proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes (Bonillo, 2006). Estas características proveen la posibilidad de crear interfaces de capa de manera estándar y a partir de ellas construir extensiones o prestaciones específicas.

3.3.3 Arquitectura Orientada a Objetos (Arquitectura OO)

Los componentes de este estilo son los objetos, o más bien instancias de los tipos de dato abstractos. En la caracterización clásica de David Garlan y Mary Shaw, los objetos representan una clase de componentes que ellos llaman managers, debido a que son responsables de preservar la integridad de su propia representación. Un rasgo importante de este aspecto es que la representación interna de un objeto no es accesible desde otros objetos.

Si hubiera que resumir las características de las arquitecturas OO, se podría decir que los componentes del estilo se basan en principios OO: encapsulamiento, herencia y polimorfismo. Son asimismo las unidades de modelado, diseño e implementación, y los objetos y sus interacciones son el

centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación. En cuanto a las restricciones, puede admitirse o no que una interfaz pueda ser implementada por múltiples clases. En tanto a componentes, los objetos interactúan a través de invocaciones de funciones y procedimientos. Hay muchas variantes del estilo; algunos sistemas, por ejemplo, admiten que los objetos sean tareas concurrentes; otros permiten que los objetos posean múltiples interfaces. Se puede considerar el estilo como perteneciente a una familia arquitectónica más amplia, que algunos autores llaman Arquitecturas de Llamada-y-Retorno (Call-and-Return) (Bonillo, 2006).

3.3.4 Arquitecturas Basadas en Componentes

Hay un buen número de definiciones de componentes, pero Clemens Alden Szyperski proporciona una que es bastante operativa: “Un componente de software, dice, es una unidad de composición con interfaces especificadas contractualmente y dependencias del contexto explícitas.” Que sea una unidad de composición y no de construcción quiere decir que no es preciso confeccionarla: se puede comprar hecha, o se puede producir en casa para que otras aplicaciones de la empresa la utilicen en sus propias composiciones (Bonillo, 2006).

En esencia, un componente es una pieza de código pre elaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar. Los componentes son los "ingredientes de las aplicaciones", que se juntan y combinan para llevar a cabo una tarea (Terreros, 2011). Las arquitecturas basadas en componentes consisten en una rama de la Ingeniería de Software en la cual se trata con énfasis la descomposición del software en componentes funcionales. Esta descomposición permite convertir componentes pre-existentes en piezas más grandes de software.

La estructura de la arquitectura basada en componentes contempla tres partes:

- El nombre de los componentes: el nombre de un componente debe ser la identificación de la funcionalidad y uso que tiene como software. Generalmente, los desarrolladores usan algún tipo de convención que facilite la identificación de componentes, especialmente, cuando se trabaja en proyectos de gran envergadura.

- La interface de los componentes: es el área de intercambio (*input-output*) entre el interior y el exterior de un componente de software. La interface es quien permite acceder a los datos y funcionalidades que estén especificadas en el interior del componente (acceder

funcionalmente, no a su especificación). Adicional a la interface se encuentra la documentación que muestra la información sobre cómo utilizar un componente.

- Cuerpo y código de implementación: es la parte del componente que provee la forma (implementación) sobre la cual un fragmento del componente realiza sus servicios y funcionalidades. Este es el área que debe cumplir con el principio de encapsulación.

La encapsulación se refiere a la especificación interna oculta o no investigable a través de la interface. Así se protege que el resto de los componentes y piezas de software dentro de un sistema, no se vean afectados por cambios en el diseño de uno de los componentes (Gil, 2011).
Las arquitecturas basadas en componentes están encaminadas a la reutilización de código mediante un diseño centrado en interfaces y componentes; en el desarrollo de los sistemas de software que siguen este estilo, una vez definidas las interfaces de comunicación de los componentes, se pueden realizar las actualizaciones e incorporación de funcionalidades de manera fácil, pues los componentes funcionan como cajas negras de las que solo se necesita saber la funcionalidad que realizan. Para efectuar cualquier cambio en el modo en su funcionamiento solo es necesario reescribir el código interno de la implementación.

3.3.5 Arquitectura Orientada a Servicios

Las Arquitecturas Orientadas a Servicios establecen un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante Servicios Web, una tecnología basada en estándares e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas en un conjunto de servicios e implementar esta funcionalidad en forma modular (Microsoft Corporation, 2006). También se puede construir una SOA empleando la invocación de métodos remotos (RMI). Las aplicaciones RMI normalmente comprenden dos programas separados: un servidor y un cliente. Una aplicación servidor típica crea un montón de objetos remotos, hace accesibles unas referencias a dichos objetos remotos, y espera a que los clientes llamen a estos métodos u objetos remotos. Una aplicación cliente típica obtiene una referencia remota de uno o más objetos remotos en el servidor y llama a sus métodos. RMI proporciona el mecanismo por el que se comunican y se pasan información del cliente al servidor y vice versa (Torrijos). El estilo arquitectónico SOA permite el desarrollo de aplicaciones distribuidas para entornos colaborativos, donde aplicaciones clientes hacen uso de las funcionalidades que mediante servicios publican aplicaciones servidoras, independientemente de tecnologías y de plataformas, es posible además que las aplicaciones asuman

ambos roles, cliente y servidor.

Se identifica SOA como parte de los elementos necesarios para desarrollar la solución que conlleva a la presente investigación, esta arquitectura ofrece además las siguientes ventajas:

- Posibilidad de reaccionar rápidamente a los cambios que se produzcan en el mercado. Dado que SOA conecta a proveedores y clientes de forma directa y transparente, agilizando toda transacción y optimizando los resultados finales, también se acelera el servicio al cliente, anticipándose la empresa a sus necesidades y reaccionando ante los cambios, siempre imprevisibles, que surjan en la demanda.

- Definición de procesos y modelos de negocio entre compañías. Gracias a la reutilización de procesos de negocio, las compañías cuentan con un sistema que fomenta la actitud colaborativa y disminuye al máximo el coste total de propiedad. Las empresas pueden contar con recursos externos, sin incrementar sus inversiones, ni adquirir tecnología complementaria a la ya existente.

- Acceso a modelos de información en tiempo real, lo que facilita a una compañía el ser totalmente independiente de las ubicaciones a través de las cuales desarrolla su operativa comercial. La proliferación de las comunicaciones a través de Internet permite utilizar los procesos de negocio de terceros, reutilizar toda su inteligencia, sin tener que invertir en nuevas estructuras (Burgos, 2008).

3.4 Requisitos

3.4.1 Requisitos funcionales de software

1. *Publicar funcionalidades en forma de un servicio de comunicación:* Se debe poder publicar funcionalidades en forma de un servicio en el Gestor de Proceso de Media y en la Plataforma de Codificación e Indexación para que sean accesibles sus funcionalidades.
2. *Consumir servicios de comunicación:* Se debe poder consumir el servicio publicado por el Gestor de Proceso de Media desde la Plataforma de Codificación e Indexación para que esta envíe la información que el Gestor de Proceso de Media necesite.

3. *Cargar plugins de consumo de nodo:* Se debe poder cargar componentes que permitan al Gestor de Proceso de Media comunicarse con la Plataforma de Codificación e Indexación y con otros sistemas.
4. *Actualizar un flujo de procesos existente:* Se debe poder modificar el estado de un flujo de procesos existente por algunos de los siguientes valores de entrada “encolado”, “ejecutando”, “finalizado”, “erróneo”, o “espera”.
5. *Eliminar un flujo de procesos existente:* Se debe poder eliminar permanentemente un flujo de procesos almacenado en el Gestor de Proceso de Media.
6. *Ejecutar un flujo de procesos existente:* Se debe poder ejecutar un flujo de procesos almacenado en el Gestor de Proceso de Media.
7. *Construir un nuevo flujo de procesos:* Se debe poder crear un nuevo flujo de procesos en el Gestor de Proceso de Media.
8. *Interpretar un fichero de configuración BPEL:* Se debe poder extraer la estructura de un flujo contenida en un archivo de configuración BPEL.
9. *Crear un nuevo tipo de procesos:* Se debe poder crear un nuevo tipo de operación a partir de la información contenida en un plugins de consumo de nodo.
10. *Crear un nuevo proceso:* Se debe poder crear un nuevo proceso que pueda ser ejecutado en uno de los nodos del Gestor de Proceso de Media.
11. *Crear un nuevo tipo de nodo:* Se debe poder crear un nuevo tipo de nodo a partir de la información contenida en un plugins de consumo de nodo.
12. *Crear un nodo de forma estática:* Se debe poder crear un nuevo nodo a partir de la información contenida en un archivo de configuraciones.
13. *Crear un nodo de forma dinámico:* Se debe poder crear un nuevo nodo mediante el servicio publicado en el Gestor de Proceso de Media.
14. *Eliminar un nodo existente*: Se debe poder eliminar permanentemente un nodo existente.

15. *Actualizar un nodo existente:* Se debe poder modificar las propiedades de un nodo.
16. *Realizar una búsqueda avanzada de nodos:* Se debe poder realizar búsquedas filtrando por el estado, conexión, e hilos.
17. *Realizar una búsqueda avanzada de flujos de procesos*: Se debe poder realizar búsquedas filtrando por el estado, fecha y tipo de proceso.
18. *Buscar un nodo idóneo:* Se debe poder buscar el nodo idóneo para ejecutar un proceso.
19. *Recuperar los procesos de los nodos desconectados:* Se debe poder asignar la ejecución de un proceso que se estaba ejecutando en un nodo que se ha desconectado a otro nodo.
20. *Restaurar el gestor de procesos de media:* Se debe poder reanudar el Gestor de Proceso de Media luego de un siniestro.
21. *Detener un flujo de proceso:* Se debe poder anular la ejecución de un flujo de proceso.
22. *Permitir la inclusión de nuevas lógicas de negocios al sistema*: Se debe permitir adicionar nuevas funcionalidades al sistema en forma de componente.
23. *Codificar un video a un formato específico:* Se debe poder convertir un video a los formatos ogg, h264 y mpeg1.
24. *Codificar un audio a un formato específico:* Se debe poder convertir un video a los formatos ogg y mp3.
25. *Transferir una media de un servidor ftp a un directorio local:* Se debe poder realizar la copia de una media de un servidor ftp a un dispositivo de almacenamiento local.
26. *Transferir una media de un directorio local a otro directorio local:* Se debe poder realizar la copia de una media de un directorio local a un dispositivo de almacenamiento local.
27. *Transferir una media de un directorio local a un servidor ftp:* Se debe poder realizar la copia de una media de un directorio local a un a un servidor ftp.
28. *Transferir una media de un servidor ftp a otro servidor ftp:* Se debe poder realizar la copia de una media de un servidor ftp a un a un servidor ftp.

3.4.2 Requisitos no funcionales de software

Fiabilidad

Disponibilidad: el sistema debe estar disponible todas las horas del día. Tiempo medio de reparación: el sistema no debe estar fuera de servicio más de 2 horas a partir de un fallo.

Eficiencia

El sistema debe ser eficiente a las peticiones realizadas en cada momento, el flujo de trabajo que sigue la aplicación no permite el fallo de ninguna de las partes, pues este influye de manera drástica sobre el próximo paso.

Tiempo de respuesta

El tiempo de respuesta promedio de las peticiones realizadas al sistema no debe exceder un tiempo máximo de 5 segundos.

Software

Dependencias generales: Sistema operativo GNU/Linux, framework Qt con licencia Qt GNU LGPL v.2.1 y biblioteca libxmlrpc1 con licencia LGPL-2.1.

Dependencias del Gestor de Procesos de Media: Biblioteca libqt4-sql-sqlite, licencia LGPL-2.1.

Dependencias de la Plataforma de Codificación e Indexación: biblioteca Ffmpeg con licencia GPL/LGPL, biblioteca MEncoder con licencia GPL y biblioteca Expect con licencia GPL

Hardware

Servidor Gestor de Procesos de Media: Arquitectura 64 bits (x64), procesador Core2Duo y memoria de 2GB.

Servidor Plataforma de Codificación e Indexación: Arquitectura 64 bits (x64), procesador Core2Duo y memoria de 4GB.

3.5 Valoración crítica del diseño propuesto por el analista

Mediante el diagrama de clases se describe la estructura del sistema a través de la exposición de atributos, operaciones y las relaciones entre ellas.

3.5.1 Módulo Gestor de Procesos de Media

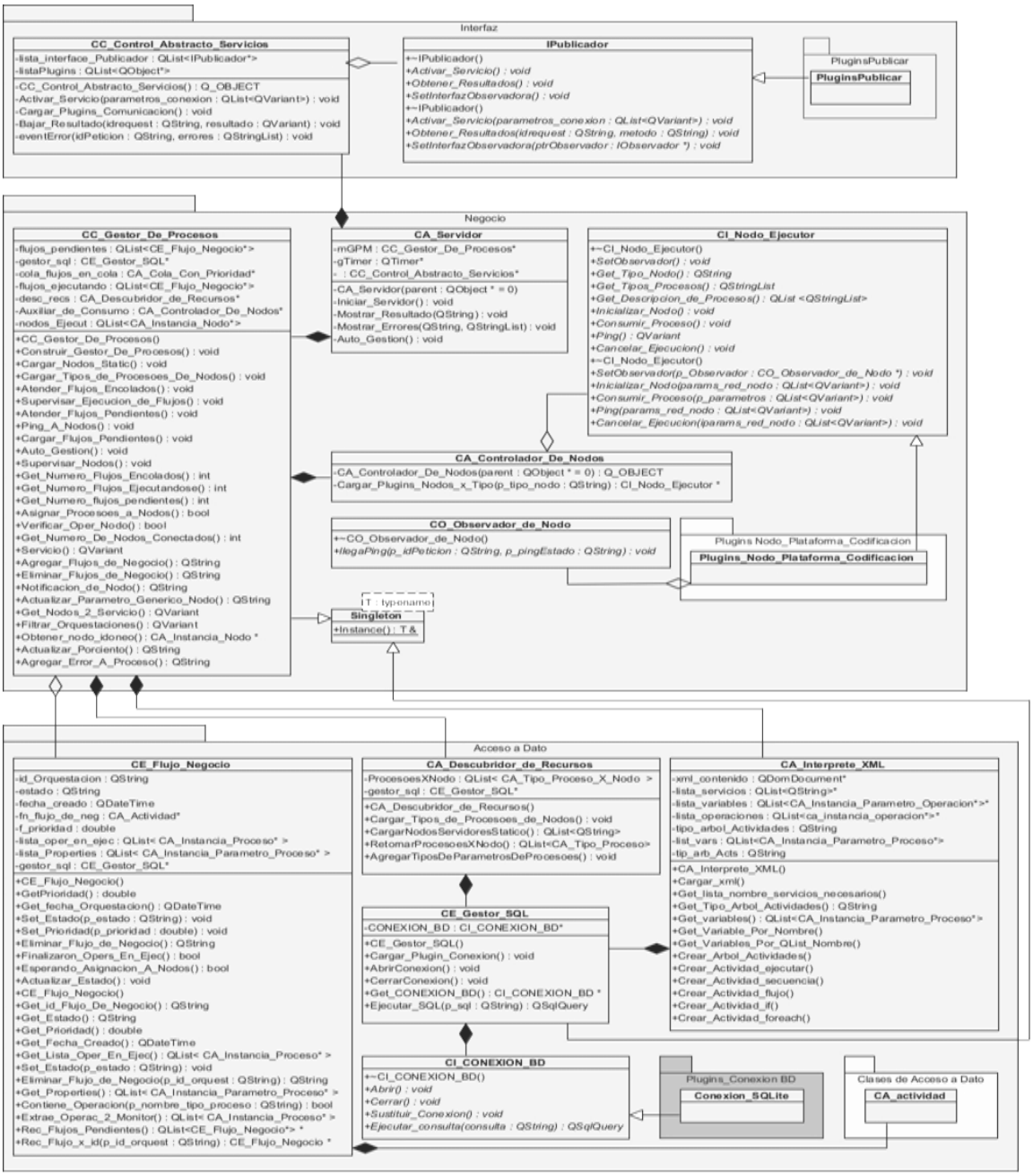

Figura 2 Diagrama de clases del diseño del Gestor de Procesos de Media.

Para lograr una adecuada organización del documento, el contenido de los paquetes “Plugins_Conexion_BD”, “Clases de Acceso a Datos” y “Plugins Nodo_Plataforma_Codificacion”, podrán ser observados en los anexos 1, 2 y 3 respectivamente.

3.5.2 Módulo Plataforma de Codificación e Indexación

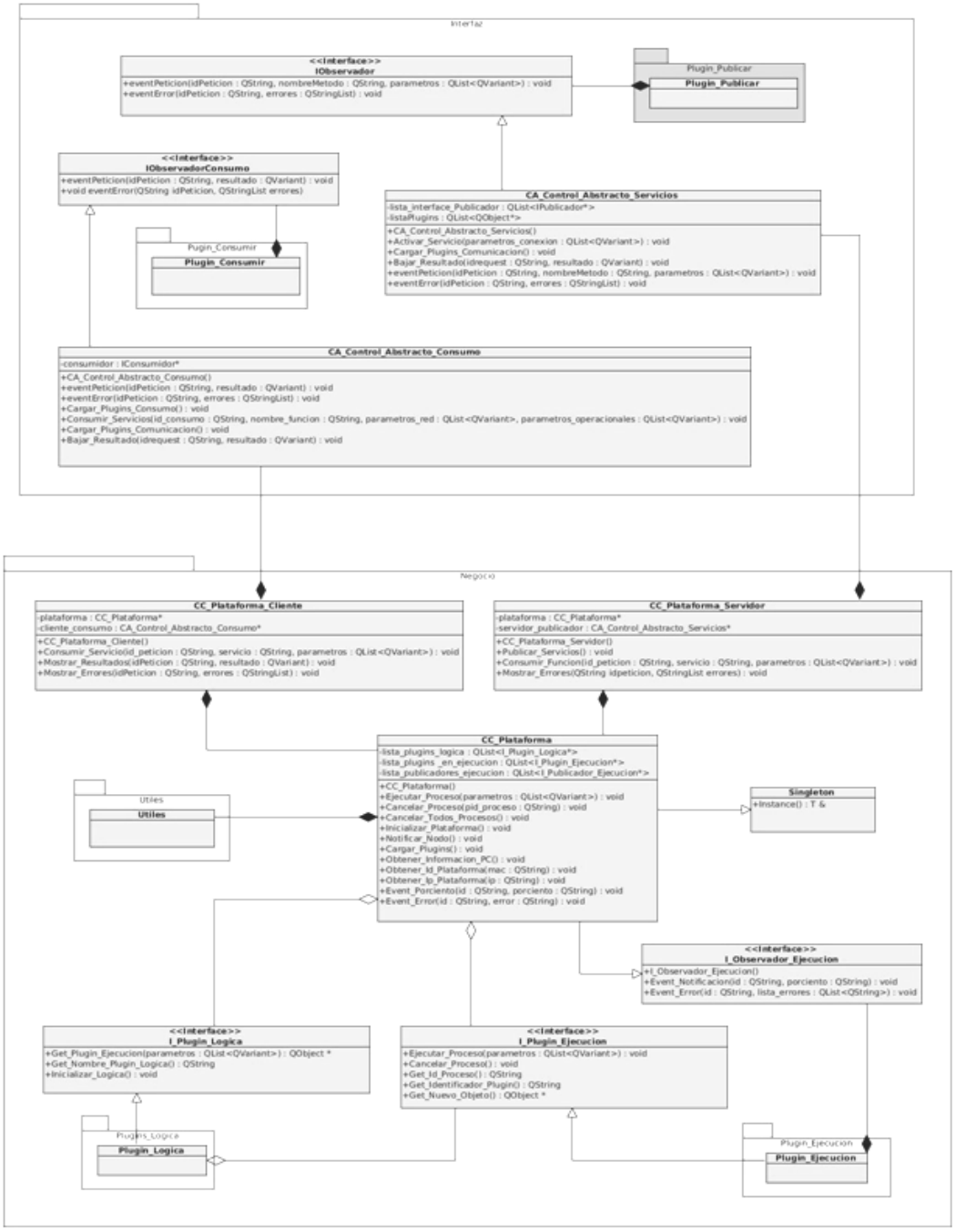

Figura 3 Diagrama de clases del diseño de la Plataforma de Codificación

3.5.2.1 Plugins Lógica

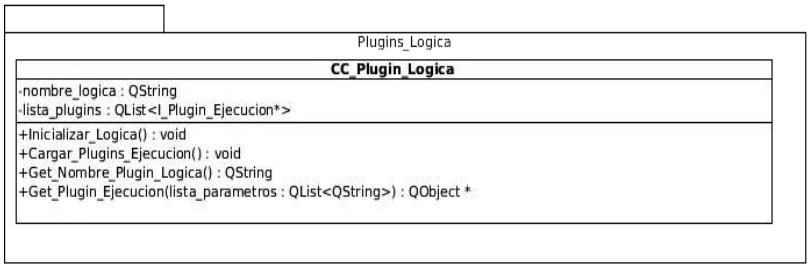

Figura 4 Diagrama de Clases del Diseño de Plugins Lógica

3.5.2.2 Plugins Ejecución

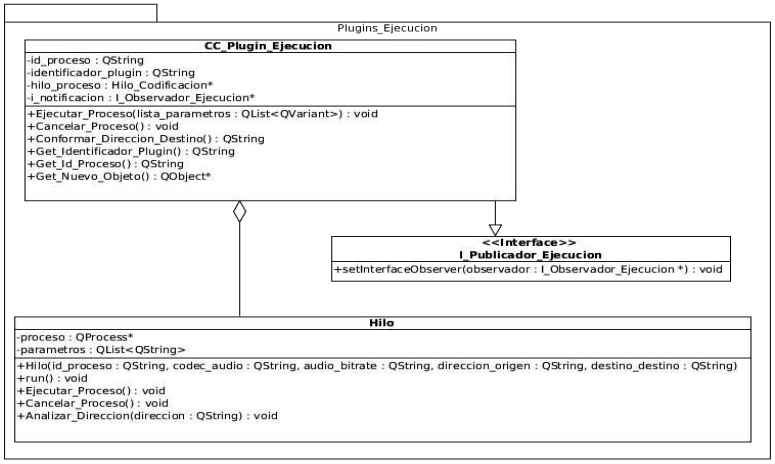

Figura 5 Diagrama de Clases del Diseño de Plugins Ejecución.

3.5.2.3 Plugins Consumir

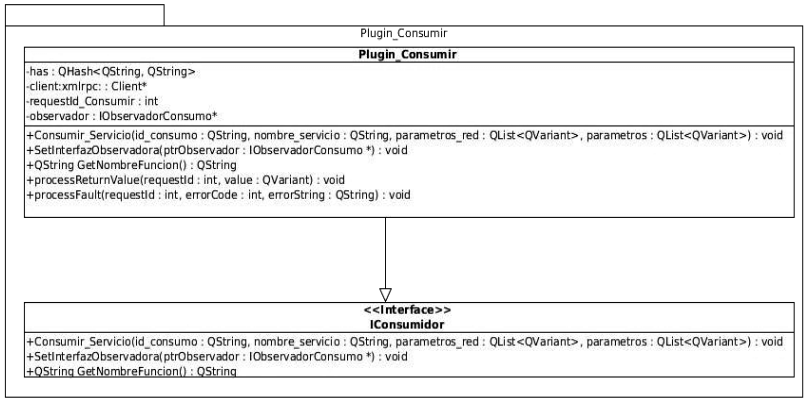

Figura 7 Diagrama de Clases del Diseño de Plugins Consumir.

3.5.2.4 Plugins Publicar

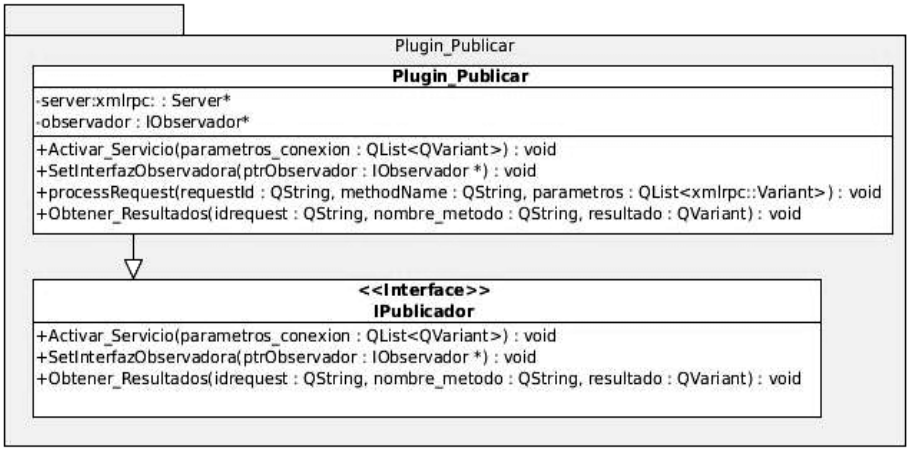

Figura 8 Diagrama de Clases del Diseño de Plugins Publicar.

3.5.2.5 Útiles

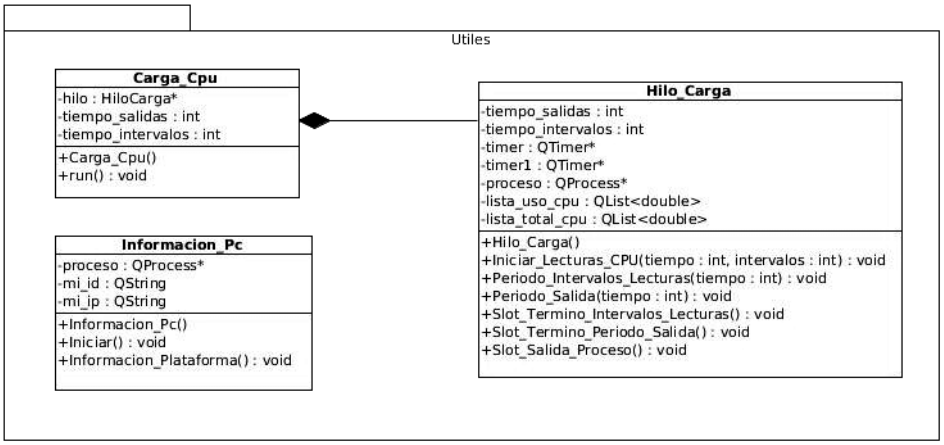

Figura 9 Diagrama de Clases del Diseño de Plugins Publicar.

En la solución propuesta por el analista las clases del diseño fueron agrupadas por módulos siguiendo el modelo arquitectónico en capas, a la par se emplean oportunamente las posibilidades brindadas por la Arquitectura Orientada a Objetos (encapsulamiento, herencia y polimorfismo) que posibilita de manera innata la reutilización de código, debido a la utilización de clases para definir objetos, y de procedimientos o métodos para definir el comportamiento de estos objetos.

Se observa a la par el uso de Arquitectura Basada en Componentes, obteniéndose finalmente un diseño altamente desacoplado. En el diagrama de clases se observa además el uso del patrón Objeto

Compuesto, el patrón Singleton y el patrón Observador, el uso de estos patrones brinda a la solución un grado alto de robustez, pues estos patrones son empleados con eficacia mundialmente para dar solución a problemas de diseño similares a los existentes en el diseño de la solución propuesta.

3.6 Diagrama Entidad-Relación del Gestor de Procesos de Media

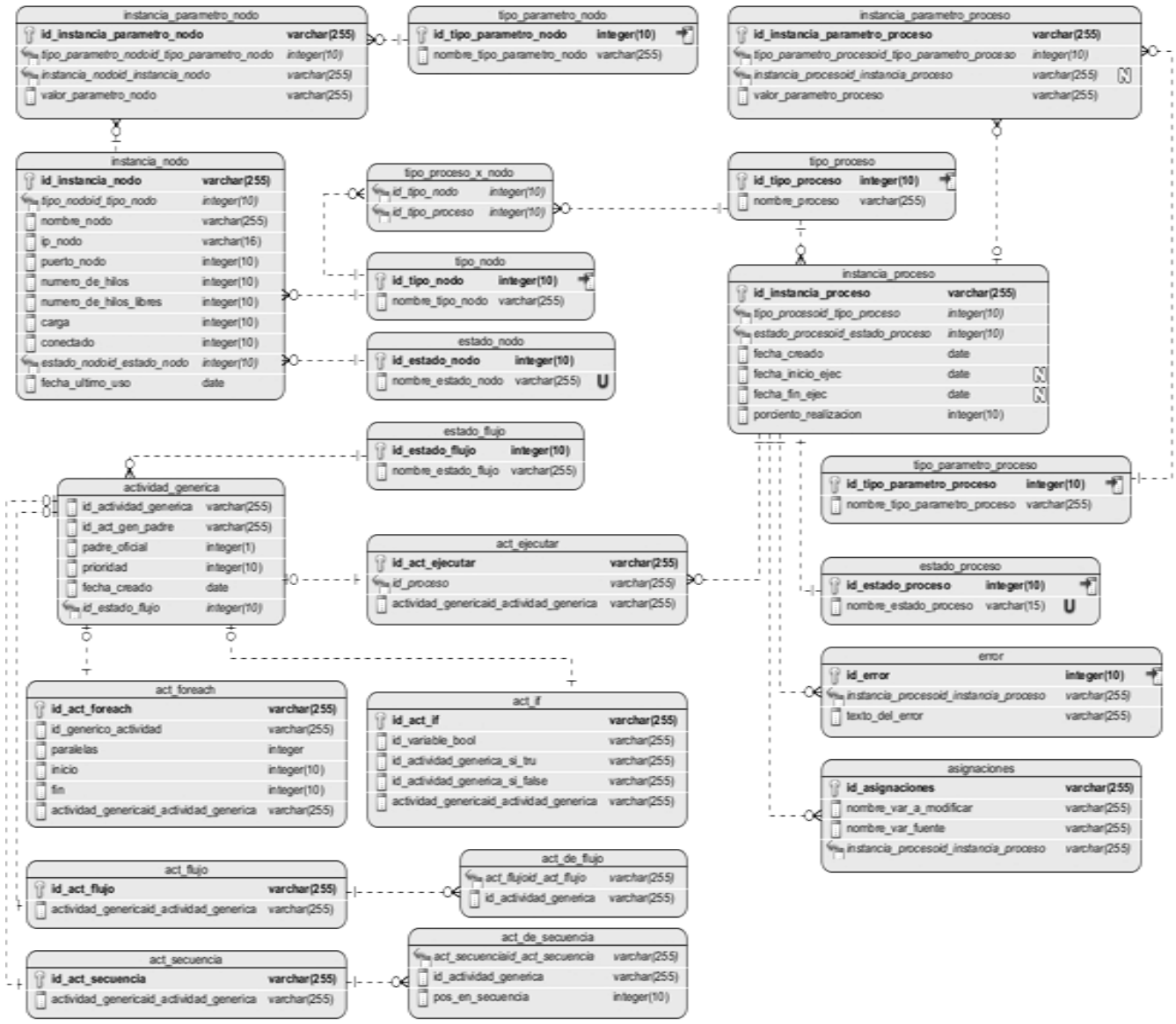

Figura 10 Diagrama Entidad-Relación del Gestor de Procesos de Media v2.

El diagrama entidad-relación muestra el diseño de la base de datos del sistema. La base de datos del Sistema Gestor de Procesos de Media v2 debe permitir el almacenamiento de las nuevas estructuras, que se originan al cuando se le adicionan nuevas funcionalidades a este, por esto es necesario el uso

del Parón Entidad Atributo Valor, pues satisface esta necesidad.

3.7 Modelo de Despliegue

El modelo de despliegue muestra las relaciones físicas entre los componentes de hardware y software en el sistema propuesto, y provee un modelo detallado de cómo se van a desplegar los componentes en la infraestructura del sistema.

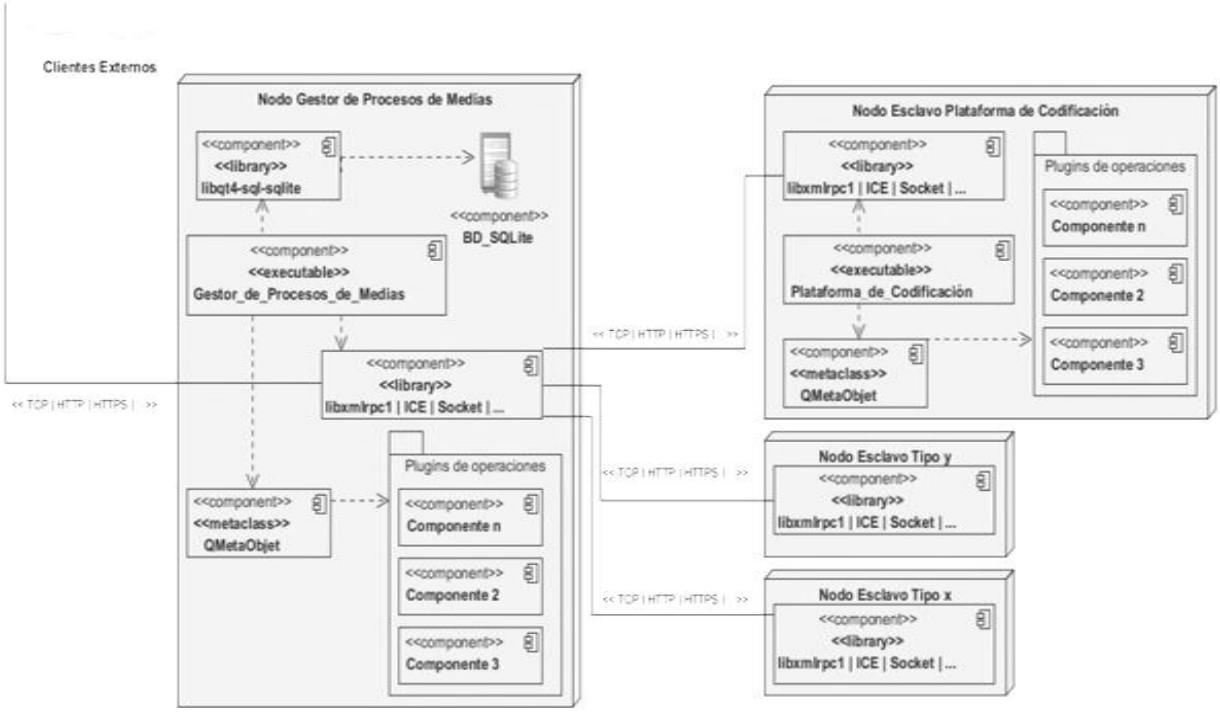

Figura 11 Diagrama Modelo de Despliegue de Sistema Gestor de Procesos de Media v2.

Se describe el Diagrama Modelo de Despliegue de Sistema Gestor de Procesos de Media v2, para una mejor comprensión de la solución a implementar.

3.7.1 Descripción de componentes

3.7.1.1 Nodo Gestor de Procesos de Media

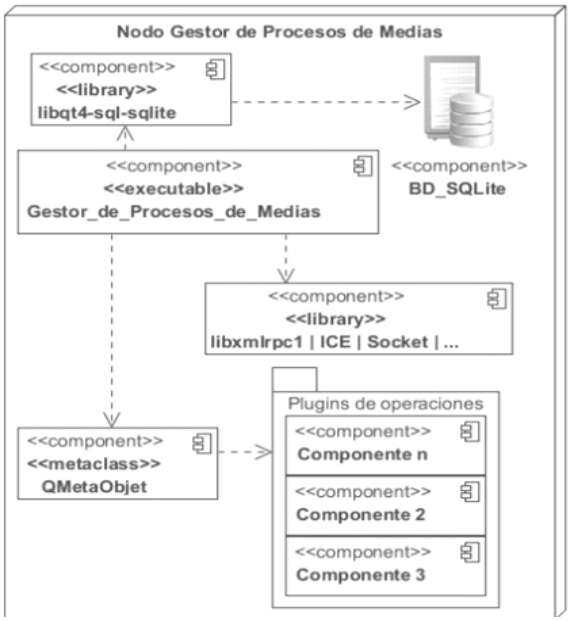

Figura 12 Nodo Gestor de Procesos de Media.

En este nodo se estará ejecutando el Gestor de Procesos de Media encargado de gestionar los flujos de procesos de medias, y la base de datos del sistema (SQLite).

3.7.1.2 Nodo Plataforma de Codificación e Indexación

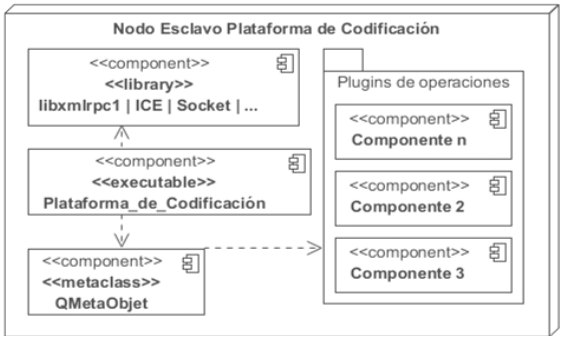

Figura 6 Nodo Plataforma de Codificación.

En este nodo se estará ejecutando la Plataforma de Codificación, encargada de realizar los procesos de codificación, indexación y transferencia sobre las medias.

3.7.1.3 Nodo Esclavo Tercero

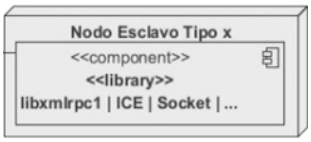

Figura 7 Nodo Esclavo Tercero.

En este nodo se estará ejecutando un sistema tercero, que se encargará de realizar un proceso específico (Ejemplo grabar video o transmitir señales), estos nodos se agregarán a la granja de nodos esclavos de gestor mediante plugins que permitan lograr la interacción con los mismos.

3.7.1.4 Clientes Externos

Figura 8 Clientes Externos.

Los clientes externos son sistemas que consumen el servicio publicado por el Gestor de Procesos de Media v2, este genera flujos de procesos descritos en configuraciones XML, los procesos resultantes serán ejecutados en los nodos esclavos según el tipo de proceso y teniendo en cuenta las capacidades de ejecución de los nodos.

3.8 Diagrama de componentes

El diagrama de componentes muestra la división del sistema en componentes y las dependencias que se generan entre los mismos.

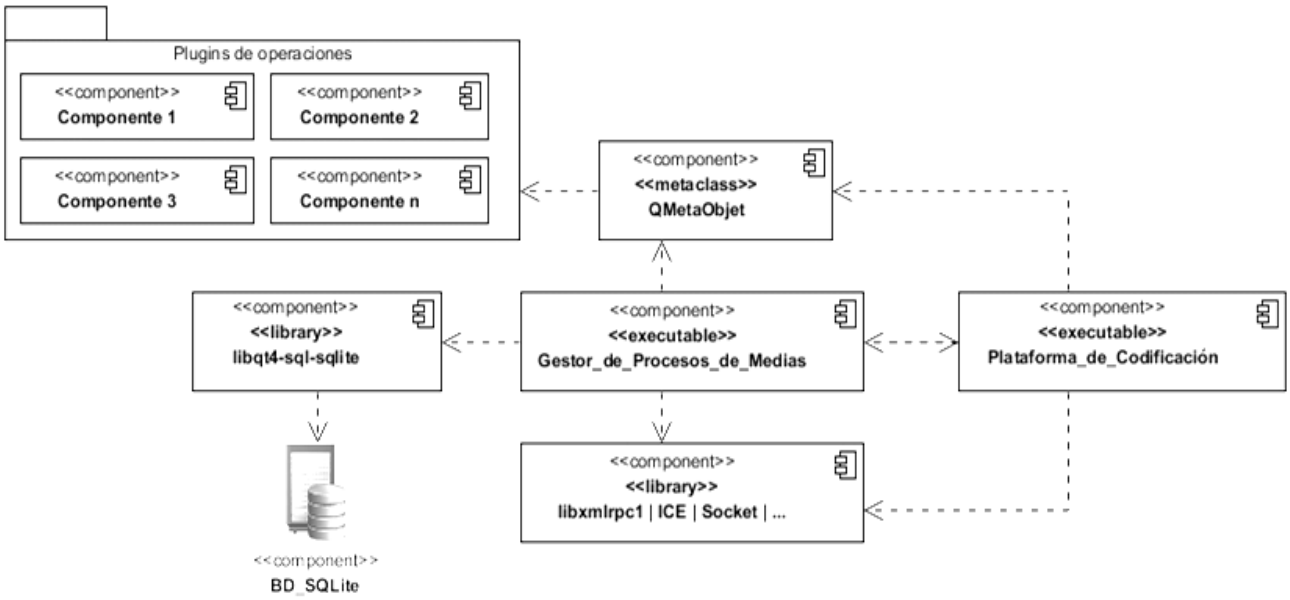

Figura 9 Diagrama de Componentes de Sistema Gestor de Procesos de Media v2.

3.9 Estilos de Programación

El estilo de programación se refiere a la forma en que se da formato al código fuente. El código de un programa lo leen muchas personas, bien para corregirlo, para ampliarlo o simplemente para evaluarlo. Para estas personas es fundamental que el código se encuentre bien redactado para que su significado sea claro y nítido. Sin embargo la legibilidad de un programa no depende sólo de las características del lenguaje sino que depende en gran medida del estilo de codificación en que está escrito. A diferencia de la sintaxis de un programa que son reglas estáticas que obligatoriamente hay que seguir, un estilo de programación está constituido por directrices que ayudan a obtener programas más legibles. Si bien no existen estilos de programación correctos o incorrectos es aconsejable la adopción de un conjunto de normas para la escritura de programas. Estas normas, básicamente, se refieren a la forma en que se colocan las llaves, a como se identa el código y como se ubican los paréntesis (Roque., 2010).

Cada programador tiene su propio estilo para escribir. Un buen estilo para programar deberá tener una estructura de código fácil de entender, no solo para otra gente sino también para sí mismo. Algunos de los elementos para lograr un buen estilo de programación, según el sitio Canal Visual Basic .net son:

- **Nombres significativos.** Los nombres de funciones y variables definidos por el programador deben ser significativos y auto-explicativos con respecto a su función. Ejemplo: Si se debe

designar una variable que almacenará la edad de una persona una forma correcta de hacerlo sería: $edad = $persona->getEdad(); y no $x = $a->getEdad();

- **Identación y espacio apropiado en el código.** La identación es usada para tener una mejor visibilidad en el diseño de un programa. La identación muestra las líneas que están subordinadas a otras líneas. Por ejemplo, todas las líneas que forman el cuerpo de un ciclo deberán estar identadas con la instrucción principal del ciclo

- **Documentando el Código.** Las complicadas e inusuales secciones de código deberán ser documentadas. Idealmente cada variable y arreglo deberá tener comentarios donde se definan tal que su función pueda ser entendida después.

- **Procedimientos Coherentes.** Cada procedimiento deberá ser diseñado para una tarea simple. Si un procedimiento maneja muchas tareas, es lógico que pueda ser difícil de entender y pueda ocurrir fácilmente un error.

- **Minimizar el acoplamiento.** Pasar un parámetro es una buena práctica de programación, pero muchos parámetros y el código pueden llegar a ser muy difíciles de manejar. Los procedimientos con muchos parámetros son altamente acoplables, ellos tienen muchas ligas u otros procedimientos. Hasta donde sea posible se deben minimizar estas ligas. Sin embargo las variables globales no deberán ser usadas en lugar de ellas.

- **Minimizar alcance de los datos hasta donde sea posible.** Las variables y los arreglos pueden ser accesados por código en diferentes partes de un programa, desde cualquier lugar si son Globales. Esto es lo ideal, pero sin no hay cuidado algunos efectos extraños pueden ocurrir en otras partes del programa, como colocando un valor en una variable por error. Restringiendo el rango de acceso, o el alcance de una variable o un arreglo se puede evitar este problema. Un alcance intermedio es el acceso necesario para una variable durante todo el simple módulo. Esto significa que cualquier procedimiento en el módulo puede tener acceso la variable, pero los procedimientos en otros módulos tienen acceso denegado (Manuales de programación, 2012).

3.9.1 Estilo K&R y BSD KNF

Estilo muy usado en el lenguaje C y PHP. Se trata de abrir la llave en la misma línea de declaración de la orden, identando los siguientes pasos al mismo nivel que la llave y cerrando la llave en el mismo

nivel que la declaración. Normalmente las tabulaciones en Windows son de 4 espacios, cuando las tabulaciones tienen 8 espacios se trata del estilo BSD o KNF, muy usado en Linux.

```
function Ejemplo($val){
	if($val == 1){
		echo $foo;
	}else {
		echo $bar;
	}

}
```

La ventaja de usar este estilo es que las llaves iniciales no necesitan ninguna línea extra para ellas solas, por lo que se ahorra espacio vertical de lectura. La desventaja de este estilo es que las llaves de cierre si necesitan una línea exclusiva para ellas. Resulta difícil identificar el comienzo de las llaves de un bloque. Sin embargo es fácil identificar el comienzo de un bloque debido a la identación a la derecha.

3.9.2 Estilo Allman

Se trata de crear una nueva línea para las llaves, e identar el código debajo de ellas. La llave de cierre tiene el mismo identado que la de inicio.

```
function Ejemplo($val)
{
	if($val % 2 == 0)
	{
		echo “El valor es par”;
	}
	else
	{
		echo “El valor es impar”;
	}
}
```

Este estilo mantiene un código limpio y claro. Las llaves de inicio y fin coinciden en la misma columna, haciendo más fácil la identificación de cada bloque. Además, es más difícil olvidarse cerrar una llave cuando se identifica claramente la llave inicial.

3.9.3 Estilo GNU

Parecido al estilo Allman, se trata de poner las llaves en una nueva línea. La diferencia es que las

llaves deben estar identadas por 2 espacios y el código dentro de ellas debe estarlo por 2 espacios más.

Para la implementación del Sistema Gestor de Procesos de Media v2 se ha utilizado el estilo Allman, teniendo en cuanta demás los elementos expuestos al inicio del epígrafe, para lograr un buen estilo de programación.

3.10 Estándar de Codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, este debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente (Microsoft, 2003).

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. Aunque la legibilidad y la mantenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas (Microsoft, 2003).

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y, posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener (Microsoft, 2003).

La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final

del proyecto de software. No es práctico, ni prudente imponer un estándar de codificación una vez iniciado el trabajo.

En la implementación del Sistema Gestor de Procesos de Media v2 se utilizará el estándar de codificación definido para c++ en el departamento de Señales Digitales, el mismo esta descrito en el Anexo No 1.

3.11 Conclusiones Parciales

Una vez realizada la valoración del diseño propuesto, se puede concluir que el diseño que se propone está bien estructurado facilitando en gran medida las bases para la implementación. Otro elemento importante tratado en este capítulo fue el estándar de codificación que se usará en la implementación del Sistema Gestor de Procesos de Media v2, elemento este importante a la hora de documentar y dar mantenimiento al sistema.

CAPÍTULO 4: Validación de la solución propuesta

4.1 Introducción

En este capítulo se exponen los resultados de la validación del Gestor de Procesos de Media v2, mediante las pruebas de Integración y de Sistema, específicamente de estas últimas las de estrés y carga. El objetivo en este capítulo es la descripción de los resultados arrojados al realizar las pruebas anteriormente mencionadas, para comprobar las funcionalidades del Gestor de Procesos de Media v2.

4.2 Proceso de Pruebas

La Real Academia Española define Prueba como la razón, argumento, instrumento u otro medio con que se pretende mostrar y hacer patente la verdad o falsedad de algo / Someterlo a determinadas situaciones para averiguar o comprobar sus cualidades o comportamientos / Ensayo o experimento que se hace de algo, para saber cómo resultará en su forma definitiva / Para referirse a lo que por su perfecta construcción, firmeza y solidez, es capaz de resistir (ESPAÑOLA, 2011).

Según Roger Pressman existen dos formas de probar cualquier producto construido, una si se conoce la función específica para la que se diseñó el producto, en ese caso se aplican pruebas que demuestren que cada función es plenamente operacional, mientras se buscan los errores de cada función, otra si se conoce el funcionamiento interno del producto se puede aplicar pruebas que aseguran que las operaciones internas se realizan de acuerdo con las especificaciones (Pressman, 2005).

Caja Blanca

Las pruebas de caja blanca se realizan si se tiene conocimiento del funcionamiento interno del producto. Permiten verificar que las operaciones internas se realizan de acuerdo a las especificaciones planteadas y que todos los componentes internos se hayan probado de manera adecuada.

Caja Negra

Las pruebas de caja negra o de comportamiento se centran en los requisitos funcionales del software. Permite derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa, es un enfoque que tiene probabilidades de errores de comportamiento o desempeño, funciones incorrectas o faltantes y en estructuras de datos a acceso a bases de datos.

Para la validación de la presente investigación se decide utilizar la técnica de prueba de Caja Negra, con la estrategia de prueba del software y las pruebas del sistema que se describen a continuación:

4.2.1 Estrategia de Prueba del software

Una estrategia de prueba del software debe servir de guía para medir los avances y buscar que los problemas aparezcan lo antes posible, esta debe incluir pruebas a nivel de componente de forma que se pueda confirmar la correcta implementación del código y que se validen las principales funcionalidades establecidas.

4.2.1.1 Integración

La Integración se basa en la búsqueda de fallas en llamadas a operaciones o conexiones entre mensajes, en este contexto pueden existir fallas de un resultado inesperado, operación incorrecta/mensaje empleado o invocación incorrecta. Determina las fallas a medida que se invocan las operaciones y examina el comportamiento de estas. Es aplicada a los atributos y a las operaciones para comprobar si ocurren valores apropiados para distintos tipos de comportamiento de objetos.

4.2.1.2 Integración Incremental Ascendente

La integración incremental se aplica de forma que el programa se construye y prueba en pequeños incrementos, lo cual permite aislar y corregir los errores que surjan. La integración Incremental ascendente, prueba seleccionada en la presente investigación, comienza con la construcción y prueba de módulos atómicos. Los componentes se integran desde los niveles más bajos hacia arriba, de forma que los módulos de bajo nivel se combinen en grupos que realicen una función del software y se prueba el grupo (Pressman, 2005).

Durante el desarrollo de los módulos Gestor de Procesos de Media v2 y Plataforma de Codificación e Indexación se fueron probando cada uno de los componentes y partes funcionales de los módulos que se iban culminando de implementar, acciones que respondían a la estrategia de prueba de software seleccionada (Integración Incremental Ascendente) para comprobar el correcto funcionamiento del Sistema Gestor de Procesos de Media v2 una vez terminado.

4.2.1.3 Integración entre Módulos del Sistema

Una vez concluida la implementación de los módulos y comprobado el correcto funcionamiento de

estos por separado, corresponde dar paso a una prueba de integración donde se pueda constatar el funcionamiento del sistema como un todo. Las funcionalidades críticas a comprobar son, que la Plataforma de Codificación e Indexación al ejecutarse se notifique satisfactoriamente al Gestor de Procesos de Media para formar parte de sus nodos esclavos. Otra funcionalidad crítica que es necesaria comprobar es la correcta asignación de un proceso por parte del Gestor de Proceso de Media a la Plataforma de Codificación e Indexación para que esta lo ejecute, de igual forma es necesario comprobar que una vez que se realice una asignación la Plataforma de Codificación e Indexación comunique al Gestor de Procesos de Media el porciento de ejecución del proceso asignado, o en caso contrario informe el error correspondiente, los resultados de la prueba se describen a continuación:

Se ejecutó el Gestor de Procesos de Media, desde una aplicación cliente se realizó una petición hacia el Gestor de Procesos de Media para crear un flujo de procesos, en dicho flujo uno de los procesos era una codificación de video, dicho proceso puede ser ejecutado en una Plataforma de Codificación e Indexación. Luego se ejecutó una Plataforma de Codificación e Indexación configurada para ser nodo del Gestor de Procesos de Media, esta se notificó correctamente con el Gestor de Procesos de Media, acto seguido este último le asignó el proceso de codificación. Tras unos instantes la Plataforma de Codificación e Indexación comenzó a enviar los porcientos referentes a la ejecución del proceso asignado. Al llegar el porciento número cien, se dejaron de enviar porcientos, para una culminación exitosa de la prueba.

4.2.1.4 Integración con terceros

El Sistema Gestor de Procesos de Media debe poder asignar ejecución de procesos a nodos que no sean del tipo Plataforma de Codificación e Indexación, para probar esta funcionalidad se prueba integrar al Gestor de Procesos de Media una aplicación existente en el departamento de Señales Digitales que graba video (Grabador) y publica esta funcionalidad mediante un servicio en la red. Es necesario recordar que este sistema no es capaz de notificarse al Gestor de Procesos de Media, ni de enviar porcientos, ni errores. Para lograr la integración del Sistema Gestor de Procesos de Media v2 con sistemas terceros, se desarrolla un componente para el Gestor de Proceso de Media, este componente permite al Gestor de Procesos de Media tratar al Grabador como un nodo más de su granja de servidores. La prueba para comprobar la correcta integración entre el Sistema Gestor de Procesos de Media v2 y el Grabador es descrita a continuación:

Se configuró un Gestor de Procesos de Media para que al iniciarse agregara automáticamente al nodo

Grabador a su granja de nodos esclavos, se ejecutó el Gestor de procesos de Media y este automáticamente agregó al nodo grabador a su granja de nodos. Desde una aplicación cliente se realizó una petición hacia el Gestor de Procesos de Media para crear un flujo de procesos, en dicho flujo uno de los procesos era una Grabación de video, dicho proceso solo puede ser ejecutado en un Grabador. El Gestor de Procesos de Media asignó el proceso de grabación al nodo Grabador y haciendo uso del componente antes mencionado mantiene un control del porciento de realización y del estado del proceso de grabación. Obteniéndose de esta forma los resultados satisfactorios esperados de esta prueba.

4.2.2 Pruebas del Sistema

Las pruebas de Rendimiento, prueban el rendimiento del software en el tiempo de ejecución. Ayudan a mejorar las capacidades de una aplicación observando y evaluando las respuestas del sistema ante todas las posibles situaciones que se puedan dar. Cada una de estas pruebas tiene sus propios objetivos y características. Para validar los resultados obtenidos a los problemas que se persiguen solucionar, esta investigación se centrará en las pruebas de carga.

Pruebas de carga: Esta prueba se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación (Testhouse Consultores S.A., 2010).

4.2.3 Prueba de carga

El Gestor de Procesos de Media ha sido integrado con la Plataforma de Codificación Indexación. Se realizaron pruebas de carga al Gestor de Procesos de Media para poder observar el comportamiento de los recursos del computador donde se encuentra ejecutándose. En un entorno real donde sea desplegada esta solución, se deben desplegar solamente cuatro estaciones donde existan aplicaciones clientes que consuman el servicio publicado por Gestor de Procesos de Media desplegado, para monitorizar los flujos de procesos y los nodos esclavos de este.
Para las pruebas se colocaron seis Monitores de Procesos distribuidos en varios computadores y se agregaron 6 nodos Plataforma de Codificación Indexación a la granja de servidores, cada uno de estos en distintos puntos de red. Se realizaron peticiones para crear flujos de procesos hasta el punto en que todos los nodos plataforma de codificación estaban ejecutando procesos, y notificando los respectivos

porcientos al Gestor de Procesos de Media. Se realizaron peticiones concurrentes para monitorizar flujos y nodos, estas peticiones se realizaron con intervalos de tiempo de un segundo desde estos los seis Monitores. El Gestor de Procesos de Media respondió rápidamente a cada una de las peticiones y los recursos de hardware del computador donde se encuentra ejecutándose se mantienen estable (Core 2Duo 1GB RAM).

4.3 Conclusiones Parciales

Las pruebas que se seleccionan se consideran acertadas y adecuadas pues se comprueba el buen funcionamiento de cada uno de los componentes desarrollados y se logra el correcto comportamiento de la aplicación en el ambiente para el cual fue concebido.

Conclusiones

Una vez concluida la investigación es importante destacar una serie de aspectos que se presentan a continuación:

- El diseño y las tecnologías seleccionadas para desarrollar el sistema fueron las correctas, pues se logró obtener un sistema con las características deseadas, permitiendo además, que al emplear herramientas libres, se lograra independizar la solución de licencias propietarias costosas.

- Con el desarrollo del Sistema Gestor de Procesos de Media v2 se logró proveer al departamento de Señales Digitales de una solución de software robusta, altamente configurable e integrable con ambientes ya existentes en distintos negocios, brindando la posibilidad de satisfacer las necesidades que puedan tener los diversos clientes.

- La utilización de patrones de diseño en la implementación del Sistema Gestor de Procesos de Media v2 propicia una mejor estructura, entendimiento y un desarrollo más rápido.

- Las pruebas que se seleccionan se consideran acertadas y adecuadas pues tienen como objetivo el correcto funcionamiento de la integración del Gestor de Proceso de Media con la Plataforma de Codificación e Indexación.

Recomendaciones

- Estudiar el equipamiento y los sistemas que existen actualmente en los distintos negocios, que constituyan clientes en potencia para el departamento de Señales Digitales y desarrollar componentes que permitan la integración de estos al Sistema Gestor de Procesos de Media v2, acortando así el tiempo en satisfacer las necesidades de futuros clientes.

- Continuar el desarrollo de componentes que incrementen el número de funcionalidades del módulo Plataforma de Codificación Indexación.

Trabajos citados

Testhouse Consultores S.A. 2010. Testhouse. [En línea] 2010. [Citado el: 20 de 4 de 2012.] http://www.es.testhouse.net/pruebas-de-rendimiento.
Alvarez, Dr. Pedro Mejía. 2011. STUDIES. [En línea] 2011. [Citado el: 15 de noviembre de 2011.] http://studies.ac.upc.edu/FIB/CASO/seminaris/2q0102/T3.ppt.
Bonillo, Pedro. 2006. *METODOLOGÍA PARA LA GERENCIA DE LOS PROCESOS DEL NEGOCIO SUSTENDA EN EL USO DE PATRONES.* Caracas, Venezuela : Centro ISYS, Facultad de Ciencias, UCV, 2006. 1807-1775.
Brokken, Frank B. 2012. *C++ Annotations.* s.l. : University of Groningen, 2012.
Burgos, Carlos. 2008. *MKM,Las ventajas de SOA.* s.l. : Editorial MKM, 2008. 147.
CHACÓN, JULIO CÉSAR RUEDA. 2006. *Aplicaciñon de la metodologia RUP para el desarrollo rápido de aplicaciones basado en el estandar J2EE.* Guatemala : Universidad de San Carlos de Guatemala, 2006.
—. 2003 . *Aplicaciñon de la metodologia RUP para el desarrollo rápido de aplicaciones basado en el estandar J2EE.* Guatemala : Universidad de San Carlos de Guatemala , 2003 .
David. 2008. Definición ABC. *Definición ABC » Proceso.* [En línea] 24 de agosto de 2008. [Citado el: 17 de noviembre de 2011.] http://www.definicionabc.com/general/proceso.php.
Definición De. 2011. Definición De. *Definición de datos.* [En línea] 2011. [Citado el: 15 de noviembre de 2011.] http://definicion.de/datos/.
definicion.de. 2008. Definicion.de. *Concepto de gestión.* [En línea] Definicion.de, 2008. [Citado el: 12 de 10 de 2011.] http://definicion.de/gestion/.
Definiciones. 2007. Definiciones. *Definiciones.com.* [En línea] 2007. [Citado el: 17 de noviembre de 2011.] http://www.definiciones.com.mx/definicion/C/codificar/.
EcuRed. 2012. EcuRed. *Conocimiento con todos y para todos.* [En línea] 2012. http://www.ecured.cu/index.php/Visual_Paradigm.
Española, Real Academia. 2009. DICCIONARIO DE LA LENGUA ESPAÑOLA. [En línea] Real Academia Española, 2009. [Citado el: 13 de noviembre de 2011.] http://buscon.rae.es/draeI/SrvltObtenerHtml?LEMA=gestionar&SUPIND=0&CAREXT=10000&NEDIC=No.
ESPAÑOLA, REAL ACADEMIA. 2011. DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición. [En línea] REAL ACADEMIA ESPAÑOLA, 2011. [Citado el: 17 de noviembre de 2011.] http://buscon.rae.es/draeI/SrvltConsulta?LEMA=proceso.
—. 2011. REA ACADEMIA ESPAÑOLA. *DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición.* [En línea] 2011. [Citado el: 4 de diciembre de 2011.] http://buscon.rae.es/draeI/SrvltConsulta?LEMA=transferir.
—. 2011. REAL ACADEMIA ESPAÑOLA. *DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición.* [En línea] REAL ACADEMIA ESPAÑOLA, 2011. [Citado el: 4 de diciembre de 2011.] http://buscon.rae.es/draeI/SrvltConsulta?LEMA=codificar.
—. 2011. REAL ACADEMIA ESPAÑOLA. *DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición.* [En línea] 2011. [Citado el: 4 de diciembre de 2011.]
Gil, José A. San. 2011. SCRIBD. *Arquitectura Basada en Componentes.* [En línea] 2011. [Citado el: 4 de diciembre de 2011.] http://www.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes.
Harmonic Inc. 2011. Harmonic. *ProMedia Carbon.* [En línea] Harmonic Inc., 2011. [Citado el: 4 de diciembre de 2011.] http://www.carboncoder.com/promedia_carbon.html.
Hernandez, Joel Francia. 2012. MSDN, Usando el Patrón de Diseño Observer. [En línea] 2012. [Citado el: 26 de 01 de 2012.] http://geeks.ms/blogs/jorge/archive/2011/03/29/patr-243-n-observador-en-net-pattern-observer.aspx.

Hernando, Roberto. 2001. www.rhernando.net - . *Metodologías de desarrollo de software.* [En línea] 2001. [Citado el: 28 de Enero de 2012.] http://www.rhernando.net/modules/tutorials/doc/ing/met_soft.html.
Jiménez, Javier Bustos. 2010. *Balance de Carga Dinámico.* [ppt] Chile : Departamento de Ciencias de la Computación (DCC), 2010.
Lado, Raquel Trillo. 2010. *TALLER DE PATRONES.* s.l. : Universidad de Zaragoza, 2010.
Larman, Craig. 1999. *UML y Patrones. Introducción al analisis y diseño orientado a objetos.* Naucalpan de Juarez, Estado de México : Dawn Speth White, 1999. 970-17-0261-1.
Manuales de programación. 2012. Canal Visual Basic .net. [En línea] 2012. http://www.canalvisualbasic.net/manual/inicio-visual-basic/estilos-programacion/.
Microsoft Corporation. 2006. *La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real.* [PDF] s.l. : Microsoft Corporation, 2006.
Microsoft. 2003. MSDN. Revisiones de código y estándares de codificación. [En línea] 2003. http://msdn.microsoft.com/es-es/library/aa291591%28VS.71%29.aspx.
Muro, Pedro. 2010. ARP CALIDAD. *Definición de proceso.* [En línea] Pedro Muro, 4 de mayo de 2010. [Citado el: 14 de noviembre de 2011.] http://arpcalidad.com/definicin-de-proceso/.
Nokia Corporation. 2008. QT. [En línea] 2008. [Citado el: 06 de 02 de 2012.] http://qt.nokia.com/products/developer-tools/.
Norman, ANTONIO FUENTES. 2011. Computación Superior. [En línea] 2011. http://www.computacionsuperior.com.mx/index-2.html.
OASIS. 2007. *Web Services Business Process Execution Language Version 2.0.* 2007.
Orallo, Enrique Hernández. 2012 . *El Lenguage Unificado de Modelado (UML).* s.l. : ehernandez@disca.upv.es , 2012 .
Osmosislatina.com. 2007. Osmosislatina.com. *Guia de UML : Importancia de UML .* [En línea] 31 de 12 de 2007. [Citado el: 28 de Enero de 2012.] http://www.osmosislatina.com/lenguajes/uml/basico.htm.
Phylum. 2012. Phylum. *Mejores Prácticas - RUP .* [En línea] 2012. http://phylum.com.mx/es/soluciones/systems-integration-a-technology/67-metodologias.html?start=5.
PostgreSQL Global Development Group . 2012. Pstgresql. [En línea] 2012. [Citado el: 5 de Enero de 2012.] http://www.postgresql.org/.
PostgreSQL Global Development Group. 1996. PostgreSQL . [En línea] PostgreSQL Global Development Group, 1996. [Citado el: 29 de 01 de 2012.] http://www.postgresql.org/about/advantages/.
Pressman, Roger. 2005. *Ingeniería de Software. Un enfoque práctico. .* s.l. : Mc Graw Hill, 2005.
Quiñones, Ernesto. 2011. *INTRODUCCION A POSTGRESQL.* Perú : Asociación peruana de software libre.http://www.apesol.org, 2011.
Restrepo, Guillermo González. 2011. Facultad de Ingeniería - Universidad de Antioquia. *El Concepto y Alcance de la Gestión Tecnológica.* [En línea] Facultad de Ingeniería de la Universidad de Antioquia, 2011. [Citado el: 13 de noviembre de 2011.] http://jaibana.udea.edu.co/producciones/guillermo_r/concepto.html.
Reynoso, Billy. 2005. *Architect Academy: Seminario de Arquitectura de Software.* s.l. : UNIVERSIDAD DE BUENOS AIRES, 2005.
Reynoso, Carlos y Kiccillof, Nicolás. 2005. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2005.
Roque., Ing. Joel Macías. 2010. *SISTEMA DE GESTIÓN DE DATOS GEOLÓGICOS MÓDULO: REGISTRO MINERO. ROL: IMPLEMENTADOR.* Ciudad de la Habana : s.n., 2010.
SCRIBD. 2012. scribd.com. *HERRAMIENTAS-CASE.* [En línea] 2012. [Citado el: 27 de 01 de 2012.] http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.
Sommerville, Ian. 2006. *Software Engineering by Sommerville (8th Edition).* 2006.
Soto, Pedro Campos. 2008. *Estudio, Desarrollo e Implementación de Algoritmos para la Paralelización de una Biblioteca de Cálculo Científico basada en Aritmética Intervalar.* 2008.
SQLlite.org. 2012. SQLlite. [En línea] 2012. http://www.sqlite.org/about.html.

Tedeschi, Nicolás. 2012. MSDN.¿Qué es un Patrón de Diseño? . [En línea] Microsoft, 2012.
TEDIAL Tecnologías Digitales Audiovisuales, S.L. *BPM, now a reality.* Málaga, España : TEDIAL Tecnologías Digitales Audiovisuales, S.L.
TEDIAL Tecnologías Digitales Audiovisuales, S.L. *TD Indexer ® Módulo de indexación y catalogación automática.* [PDF] Malaga, España : Tedial.
Terreros, Julio Casal. 2011. MSND. *Desarrollo de Software basado en Componentes .* [En línea] 2011. [Citado el: 04 de Diciembre de 2011.] http://msdn.microsoft.com/es-es/library/bb972268.aspx.
Torrecillas, Javier Morueco. RAID - Tolerancia a Fallos. *RAID - Tolerancia a Fallos.Tecnología RAID (Redundant array of independent disks):Soluciones Tolerantes al Fallo.*
Torres, Manuel Lagos. 2008. Introducción al diseño con patrones. [En línea] 2008. http://www.elrincondelprogramador.com/default.asp?id=29&pag=articulos/leer.asp.
Torrijos, Ricard Lou. Invocación Remota de Métodos (RMI). *Programación en Castellano.* [En línea] [Citado el: 19 de noviembre de 2011.] http://www.programacion.com/articulo/invocacion_remota_de_metodos_rmi_107/2.
Universidad Carlos III de Madrid. E-Archivo. [En línea] Universidad Carlos III de Madrid. Biblioteca. [Citado el: 18 de noviembre de 2011.] http://e-archivo.uc3m.es/help/glosario.html.
Vallespir, Diego. 2011. Facultad de Ingeniería. Universidad de la República - Uruguay. [En línea] 2011. http://www.fing.edu.uy/inco/cursos/ingsoft/pis/proceso/MUM/dat/intro/intro.htm.
Visual Paradigm. 2012. Visual Paradigm. [En línea] 27 de 01 de 2012. http://www.visual-paradigm.com/product/vpuml/.
WEBNOVA. 2011. WEBNOVA. *Web Services - XML-RPC, SOAP, sobre PHP, Perl, y otros conceptos.* [En línea] 2011. http://webnova.com.ar/articulo.php?recurso=426.
Welicki, León. 2011. MSDN, El Patrón Singleton. [En línea] 2011. [Citado el: 26 de 01 de 2012.] http://msdn.microsoft.com/es-es/library/bb972272.aspx#mainSection.
XMLRPC.COM. 1999. XMLRPC.COM. [En línea] 14 de 6 de 1999. [Citado el: 9 de 01 de 2012.] http://xmlrpc.scripting.com/.

GLOSARIO DE TÉRMINOS

Ordenadores: es una máquina electrónica que recibe y procesa datos para convertirlos en información útil.

Aplicación: Es una clase de programa informático creado para facilitar al usuario un determinado tipo de trabajo. Esto lo caracteriza frente a otros programas como los sistemas operativos, las utilidades y los lenguajes de programación.

Metadatos: literalmente “sobre datos”, son datos que describen otros datos.

Televisión: es un sistema para la transmisión y recepción de imágenes en movimiento y sonido a distancia.

Video: es la tecnología de la captación, grabación, procesamiento, almacenamiento, transmisión y reconstrucción por medios electrónicos digitales o analógicos de una secuencia de imágenes que representan escenas en movimiento.

Audio: Señal, técnica o sistema electrónico relacionado con la grabación, reproducción y transmisión del sonido que normalmente se encuentra acotado al rango de frecuencias audibles para los seres humanos.