Facultad 9

PROPUESTA DE UNA ARQUITECTURA EN EL DOMINIO DE MEDIA PARA LOS PRODUCTOS DEL DEPARTAMENTO DE SEÑALES DIGITALES.

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

Autora: Liudmila de Las Mercedes Estrada Rodríguez

Tutor: Ing. Reinier Castillo González

Msc. Yaneisis Pérez Heredia

Ciudad de la Habana, 2010

"Año del 51 Aniversario del Triunfo de la Revolución"

Declaración de autoría

Declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los 26 días del mes de junio del año 2010.

_____	_____
Firma de la autora:	Firma del tutor:
Liudmila de Las Mercedes Estrada Rodríguez	Reinier Castillo González

Dedicatoria

Dedico el presente trabajo al ser más grande de mi vida, mi madre: Beatriz Rodríguez Miranda, luz, faro y guía de cada uno de mis actos. A mi segundo padre Hernán Rodríguez Cordoví quien con amor me ha ayudado a ser la persona que soy hoy. A mi padre Jorge Luis Estrada Machado , que aunque no ha estado aquí siempre ha sentido orgullo de mí. A mi abuela Enma, que aunque lejos está, siempre me acompaña.Mis hermanos, principalmente Marjita,quien tanto me ha apoyado en esta lucha, a todos mis tíos por parte de madre y padre quienes viven día a día orgullosos de llamarme universitaria y ya ingeniera, a toda mi familia en general que es bastante grande. En especial a mis grandes amigas Yordanka, Elda, Yuneysis y Roxana, quienes desde hace muchos años, han estado a mi lado en las buenas y en las más malas, quienes más que amigas para mí son mis hermanas.A los buenos amigos que he hecho a lo largo de la universidad, les agradezco su amistad y compañerismo.

Mami te agradezco todo lo que has hecho en la vida por mí,gracias por tu sacrificio de madre, por tu apoyo, tu amor, tu dedicación, gracias más que nada por regalarme la vida.

Agradecimientos

A mi madre por su amor, su cariño y la confianza que siempre depositó en mí durante todo este tiempo, enseñándome que no importa las veces que me caiga, sino las veces que me levante. A mis buenos amigos, por ser mi familia y poder contar con ellos, dejando claro que haberlos conocidos ha sido uno de los más grandes regalos que me ha dado la vida. A mi tutor Reinier Castillo González por su esfuerzo y ayudarme a seguir adelante, A mi oponente en el tribunal Yoandris Quintana, por su comprensión y ayuda.Al arquitecto René Lazo por ser mi primer guía en el desarrollo de la tesis. A Osmar Leyet por guiarme en este proceso de desarrollo de tesis. A los profesores que me impartieron clases. A los miembros del jurado por sus oportunas críticas y por la paciencia con que me han enseñado a exponer una tesis, principalmente mi jefa de tribunal Yanisley Alvarez por su calidez de enseñanza. A la revolución y a Fidel por darme la oportunidad de estudiar en esta maravillosa Universidad.

RESUMEN.

En las últimas décadas el avance de las Tecnologías de la Información y las Comunicaciones (TIC) han despertado un gran interés en las comunidades de desarrollo del mundo del cual Cuba no se encuentra ajena. Las necesidades actuales que tiene toda organización para el logro de sus objetivos, demandan la construcción de grandes y complejos sistemas de software que requieren de la combinación de diferentes tecnologías y plataformas de hardware y software para alcanzar un funcionamiento acorde con dichas necesidades. Lo anterior, exige de los profesionales dedicados al desarrollo de software poner especial atención y cuidado al diseño de la arquitectura, bajo la cual estará soportado el funcionamiento de sus sistemas. Hoy día muchos sistemas fracasan o no alcanzan el total de los requisitos establecidos porque muchas veces la arquitectura de software se encuentra deficiente en su concepto o diseño, o quizás no cuentan con la arquitectura del software adecuada para el sistema que se desea desarrollar. Este trabajo se centra hacia el desarrollo de una arquitectura robusta y confiable que cumpla con todos los requisitos necesarios para el desarrollo de futuras aplicaciones que se implementen en el Departamento de Señales Digitales.

ÍNDICE

INTRODUCCIÓN

Desde tiempos muy remotos, el hombre ha sentido la necesidad de expresar ideas. Su búsqueda constante por satisfacer cada vez mejor su necesidad de comunicación ha sido el impulso que ha logrado la instauración en el mundo de instrumentos cada día más poderosos y veloces en el proceso comunicativo. A partir del incesante avance de la ciencia y la tecnología, la comunicación dejó de ser exclusivamente oral para desarrollarse a través de otros medios. El hombre ha sentido la necesidad de saber y de obtener esta información transmitida por los medios de comunicación. En la actualidad, las sociedades industrializadas dependen, en gran medida, de los medios de comunicación masivos. Su sistema económico basado en la compraventa generalizada, la compleja división del trabajo y las necesidades del estado, para cumplir con sus funciones requieren de estos medios para difundir la información del modo más rápido y a la mayor cantidad de personas posible. De allí que cada vez sea más estrecha la relación entre los grandes grupos económicos y las grandes cadenas de comunicación.

En las últimas décadas la aparición de nuevas tecnologías ha generado la necesidad de modos de trabajo. Ha creado formas en que los sistemas de desarrollo actuales dan soporte a estructuras software complejas. Proponer entonces, un marco de referencia para el desarrollo de una arquitectura en el dominio de medias, es de gran importancia para el desarrollo de cualquier sistema. Con el desarrollo de las medias, se integrarán componentes para hacer ciertas tareas que proporcionarán a los usuarios oportunidades de trabajo y acceso a diversas tecnologías. Es un medio, donde la computadora junto con los medios tradicionales dan otra forma de expresión. Es una experiencia, donde la interacción con los medios es radicalmente diferente. Es una industria, que con una plataforma, un medio y una experiencia lleva a tener otras oportunidades de negocios.

Cuba no se encuentra ajena al uso de las medias, algunas empresas se dedican a la construcción de productos que las usan para un mejor desarrollo del país, apareciendo entornos informáticos que atienden las necesidades de los ciudadanos, abriendo esto, posibilidades de trabajo mucho más flexibles. En Cuba no solo se realizan trabajos con medias, sino que se trabaja fuertemente en la informatización de la sociedad. Un ejemplo claro de esto es la Universidad de las Ciencias Informáticas (UCI), donde se

gradúan miles de ingenieros anuales de todo el país, los cuales serán protagonistas de la informatización de la sociedad.

La UCI está compuesta por 10 facultades, cada una de ellas con sus respectivos polos productivos en los cuales trabajan un conjunto de profesores y estudiantes para el desarrollo de software. La facultad 9 desde sus inicios dedicó sus esfuerzos al trabajo televisivo, creándose con el transcurso de los años el Departamento de Señales Digitales.

El Departamento de Señales Digitales de la facultad 9 tiene varios productos, y se considera la selección de una arquitectura que establezca un marco de trabajo estructural básico para estos productos. Es preciso establecer una arquitectura que potencie el rendimiento de estos productos, de forma tal que todos puedan trabajar integrados en un entorno común. Es importante la selección de un estilo arquitectónico idóneo que facilite el proceso de desarrollo de los productos que trabajan con medias. Se ha planteado la necesidad de una arquitectura común para todos estos productos que facilite el desarrollo y el entorno de trabajo de estos productos.

A partir del estudio de la situación antes planteada se identificó como **problema científico**: Necesidad de mejorar el desarrollo de los productos del Departamento de Señales Digitales.

Atendiendo a las necesidades del departamento de Señales Digitales el **objeto de estudio es**: Arquitectura de software y se define como **campo de acción**: arquitectura de software de los productos del Departamento de Señales Digitales.

Como **Idea a defender** se tiene que si se logra un marco de referencia para el desarrollo de una arquitectura en el dominio de media para los productos del Departamento de Señales Digitales, entonces se potenciara el proceso de desarrollo y rendimiento de los mismos.

El **objetivo general** de este trabajo es proponer un marco de referencia para el desarrollo de una arquitectura en el dominio de media para los productos del Departamento de Señales Digitales.

Para lograr este objetivo se desarrollan las siguientes **tareas de la investigación:**

1. Evaluar alternativas de arquitecturas para identificar los puntos comunes en los productos del Departamento de Señales Digitales.
   1.1. Valorar el estado del arte de varios estilos y patrones de la arquitectura de software y tomar posición en cuanto al más idóneo para la propuesta.
2. Identificar los atributos de calidad de las medias en el Departamento de Señales Digitales.
3. Definir los escenarios arquitectónicos.
4. Seleccionar el/los estilos arquitectónicos apropiados para el Departamento de Señales Digitales.
5. Seleccionar los estándares para el procesamiento y transmisión de información.
6. Definir las tecnologías telemáticas y de servicios.
7. Identificar las herramientas y frameworks idóneos para el desarrollo de las medias en el Departamento de Señales Digitales.

Métodos Teóricos

Histórico-Lógico: para determinar las tendencias actuales, en este caso para realizar el estudio de algunos elementos de la arquitectura que servirán de guía para una buena arquitectura de software .

Analítico-Sintético: permite desarrollar y procesar toda la información en partes un poco más cortas para una mejor comprensión, dándole a esas pequeñas unidades un análisis más claro y concreto.

1
Capítulo

Fundamentación teórica.

CAPÍTULO 1: Fundamentación teórica.

1.1 Introducción

En este capítulo se aborda sobre el análisis del estado de arte de la arquitectura de software partiendo del estudio de numerosas arquitecturas existentes y basadas en los principales conceptos de arquitectura, diferentes estilos y patrones arquitectónicos. Se analizan además alternativas de arquitecturas para a partir de estas, proponer un marco de referencia para el desarrollo de una arquitectura en el dominio de media para los productos del Departamento de Señales Digitales.

1.2 Situación problemática

En los últimos años las Tecnologías de la Información y las Comunicaciones (TIC) han potenciado el estudio, profundización y desarrollo de nuevos productos en las comunidades científicas a nivel mundial. Cuba no está exenta de los avances que han surgido en la nueva era de la información, por lo que se hace necesario de informatizar a la sociedad cubana. En esta batalla la Universidad de las Ciencias Informáticas desempeña un papel protagónico debido a que cuenta con todos los recursos y el capital humano necesario para emprender y desarrollar proyectos informáticos con gran impacto social. La facultad 9 como partícipe de este proceso cuenta un departamento de Señales Digitales que desarrolla numerosos productos, desde aplicaciones para la industria del petróleo, hasta sistemas de transmisión de noticias. En aras de satisfacer la gran demanda que genera el mercado y las constantes peticiones de los clientes, se hace necesario establecer arquitecturas de software robustas, eficientes que potencien el rendimiento de las aplicaciones. En el Departamento de Señales Digitales se han desarrollado múltiples aplicaciones encaminadas a mejorar todo tipo de servicios informáticos. A partir de lo antes mencionado, en el presente trabajo se realizará una propuesta de la arquitectura idónea para el trabajo con las medias, con el objetivo de mejorar el proceso de desarrollo de los productos del departamento.

1.3 ¿ Qué son las medias ?

Se consideran Medias a los videos, sonidos, textos, imágenes y animaciones. Se pueden integrar en un mismo entorno haciéndolo más llamativo para el usuario, para obtener un resultado visible, audible o ambas cosas. Las medias integradas pueden proyectarse, verse en un escenario, transmitirse o reproducirse localmente en un dispositivo (19).

1.5 EJEMPLOS DE APLICACIONES DE LAS MEDIAS EN EL CENTRO DE SEÑALES DIGITALES

- Plataforma de Transmisión Abierta para Radio y Televisión
- Captura y Catalogación.
- Primicia.

1.6 Arquitectura de software.

La Arquitectura de Software es hoy en día un área extensa de investigación teórica y de formulación práctica. Comenzó siendo una abstracción descriptiva y a medida del transcurso del tiempo se fueron redefiniendo todos y cada uno de sus conceptos.

Cuando se hace referencia a Arquitectura del Software se puede arribar al concepto básico: es la estructura o estructuras de un sistema, compuesta por elementos, las propiedades visibles de estos y las relaciones que mantienen dentro de ella. (1)

Otra forma de definir la Arquitectura del Software es la de Clements donde expresa que: es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se le percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. (2.)

Por otra parte, también se puede decir que la Arquitectura del Software es el estudio de la estructura a gran escala y el rendimiento de los sistemas de software. La arquitectura de un sistema que incluye la división de funciones entre los módulos de un sistema, los medios de comunicación entre los módulos y la representación de la información compartida (3.) Establece los fundamentos para que los desarrolladores de un proyecto

de software trabajen en una línea común permitiendo alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

La Arquitectura de Software está compuesta por componentes, definiendo el ¿qué? de la aplicación, posee relaciones y restricciones que definen el ¿cómo? realizar la aplicación. Posee requisitos tanto funcionales (RF) como no funcionales (RNF) ligada más a este último, que junto a las decisiones significativas definen el ¿por qué? de las decisiones a tomar, teniendo en cuenta rendimiento, confiabilidad y disponibilidad, escalabilidad etc. Permite a los miembros del grupo de desarrollo encaminar el producto por una línea de trabajo común, logrando alcanzar los objetivos propuestos.

Aunque es grande el número de definiciones de lo que es la arquitectura de software, sí se puede afirmar que es beneficiosa para los productos que se quieran desarrollar, ya que es sin dudas un método de satisfacción de requisitos y una base esencial para la administración de procesos y la consistencia de un determinado sistema.

1.7 Estilos Arquitectónicos

La clave del trabajo arquitectónico tiene que ver con la correcta elección del estilo arquitectónico.

1.7.1 ¿Qué es un estilo arquitectónico?

Los “estilos arquitectónicos” de software son arquitecturas de software comunes, marcos de referencias arquitectónicas o clases de sistemas. Específicamente, un estilo arquitectónico va a determinar el vocabulario de componentes y conectores que puede ser usado, así como un conjunto de restricciones que pueden ser combinadas. Los estilos generalmente proveen una guía y análisis para crear una clase amplia de arquitecturas en un dominio específico donde los patrones se enfocan en solucionar los problemas más pequeños y más específicos dentro de un estilo dado.(21)

Los estilos de arquitectura se definen como las 4C:

- Componentes (Elementos)
- Conectores
- Configuraciones
- Restricciones (Constraints)

Algunos de los principales estilos arquitectónicos que se usan en la actualidad están divididos por clases de Estilos y engloban una serie de estilos arquitectónicos específicos:

1.7.2 Ejemplo de algunos estilos Arquitectónicos

- **Estilos de Flujo de Datos**
  - Tubería y filtros.
- **Estilos Centrados en Datos**
  - Arquitecturas de Pizarra o Repositorio.
- **Estilos de Código Móvil**
  - Arquitectura de Máquinas Virtuales.
- **Estilos Peer-To-Peer**
  - Arquitecturas Basadas en Eventos.
  - Arquitecturas Orientadas a Servicios (SOA).
  - Arquitecturas Basadas en Recursos.
- **Estilos de Llamada y Retorno**
  - Modelo_Vista_Controlador (MVC).
  - Arquitecturas en Capas.
  - Arquitecturas Orientadas a Objetos.
  - Arquitecturas Basadas en Componentes.

1.7.2.1 Estilos de Flujo de datos

Esta familia de estilos enfatiza la reutilización y la modificabilidad. Es apropiada para sistemas que implementan transformaciones de datos en pasos sucesivos.

- *Tuberías y Filtros* (estilo arquitectónico específico): Una tubería es una popular arquitectura que conecta componentes computacionales a través de conectores, de modo que el procesamiento de datos se ejecuta como un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. Este estilo se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada.

1.7.2.2 Estilos centrados en datos

Esta familia de estilos enfatiza la integrabilidad de los datos. Se estima apropiada para sistemas que se fundan en acceso y actualización de datos en estructuras de almacenamiento.

- *Arquitecturas de Pizarra o repositorio*: Esta arquitectura está compuesta por dos componentes principales: una estructura de datos que representa el estado actual y una colección de componentes independientes que operan sobre él. Estos sistemas se han usado en aplicaciones que requieren complejas interpretaciones de proceso de señales (reconocimiento de patrones, reconocimiento de voz).

1.7.2.3 Estilos de Llamada y Retorno

Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala.

El patrón conocido como Modelo-Vista-Controlador (MVC)

Separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres partes diferentes:

- **Modelo.** El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requisitos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
-

- **Vista.** Maneja la visualización de la información.

- **Controlador**. Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.(22).

- *Arquitectura en Capas*: En este estilo arquitectónico cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior. Al dividir un sistema en capas, cada una de ellas puede tratarse de

forma independiente, sin tener que conocer los detalles de las demás. Este sistema en capas facilita el diseño modular, en la que cada capa encapsula un aspecto concreto y permite además la construcción de sistemas débilmente acoplados, lo que significa que si se minimiza la dependencia entre capas, resulta más fácil sustituir la implementación de una capa sin afectar al resto del sistema. Este es uno de los patrones que aparecen con mayor frecuencia o por el contrario, como una de las posibles encarnaciones de algún estilo más envolvente. En la práctica, las capas suelen ser entidades complejas y están compuestas por varios paquetes o subsistemas.

- *Arquitectura Orientada a Objetos:* Los componentes de este estilo son los objetos o más bien instancias de los tipos de datos abstractos. En la caracterización clásica de David Garlan y Mary Shaw (4). Los objetos representan una clase de componentes que ellos llaman managers, debido a que son responsables de preservar la integridad de su propia representación. Los componentes de esta arquitectura se basan en principios Orientados Objetos: encapsulamiento, herencia y polimorfismo. Un rasgo importante de este aspecto es que la representación interna de un objeto no es accesible desde otros objetos. Son las unidades de modelado, diseño e implementación. 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 tantos componentes, los objetos interactúan a través de invocaciones de funciones y procedimientos.

- *Arquitectura basada en Componentes:* Los sistemas de software basados en componentes se basan en principios definidos por una ingeniería de software específica. Los componentes son las unidades de modelado, diseño e implementación, las interfaces están separadas de las implementaciones y conjuntamente con sus interacciones son el centro de incumbencias en el diseño arquitectónico. Los componentes soportan algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución.

1.7.2.4 Estilo de Código Móvil

Esta familia de estilos enfatiza la portabilidad. Ejemplos de la misma son los intérpretes, los sistemas basados en reglas y los procesadores de lenguaje de comando.

- *Arquitectura de Máquinas Virtuales:* Esta arquitectura se conoce como intérpretes basados en tablas o sistemas basados en reglas. Estos sistemas se representan mediante un seudo-programa a interpretar y una máquina de interpretación. Estas variedades incluyen un extenso espectro que está comprendido desde los llamados lenguajes de alto nivel hasta los paradigmas declarativos no secuenciales de programación.

1.7.2.5 Estilos Peer _To_ Peer

Esta familia se conoce también como componentes independientes, enfatiza la modificabilidad por medio de la separación de las diversas partes que intervienen en la computación. Consiste por lo general en procesos independientes o entidades que se comunican a través de mensajes.

- *Arquitecturas Basadas en Eventos:* Estas se han llamado también arquitecturas de invocación implícita. Se vinculan con sistemas basados en publicación-suscripción. La idea dominante en la invocación implícita es que en lugar de invocar un procedimiento en forma directa, un componente puede anunciar mediante difusión uno o más eventos.

- *Arquitecturas Orientadas a Servicios (SOA en inglés):* Esta construye toda la topología de la aplicación como una topología de interfaces, implementaciones y llamados a interfaces. Es una relación entre servicios y consumidores de servicios, ambos lo suficientemente amplios como para representar una función de negocio completa.

¿Para qué se utiliza SOA?

- **Abstracción del lenguaje:** Antes de que surgiera SOA los desarrolladores hablaban de interfaz, clases, tablas, consultas y los directivos de negocio sólo hablaban de procesos de negocio y para poder lograr un entendimiento entre ellos era muy difícil. Con SOA un servicio de negocio para ambas partes es lo mismo. La posibilidad de que ambas partes tantos los directivos de negocio como los desarrolladores hablen un lenguaje común posibilita que puedan entenderse y tomar decisiones juntos.

- **Abstracción de tecnología:** Las empresas actuales están muy vinculadas a la habilidad que esta posea para adaptar sus tecnologías a los frecuentes cambios y desafíos del negocio. Una SOA permite que las tecnologías evolucionen a la par del negocio.

- **Flexibilidad de la funcionalidad:** Los bloques que se componen para armar una solución y responden a las necesidades del cliente. Al observar la estructura antigua del software, se asemeja a un bloque armado, funcional pero estático.

- *Arquitecturas Basadas en Recursos:* Esta define recursos identificables y métodos para acceder y manipular el estado de esos recursos. El caso de referencia es nada menos que la World Wide Web, donde los URIs identifican los recursos y HTTP es el protocolo de acceso. El argumento central es que HTTP mismo, con su conjunto mínimo de métodos y su semántica simplísima, es suficientemente general para modelar cualquier dominio de aplicación.

1.8 Patrones

1.8.1 Definición de Patrones

Los desarrolladores con experiencia tienen un repertorio tanto de principios generales como de soluciones basadas en aplicar ciertos estilos que les guían en la creación de software. Estos principios y estilos, si se codifican con un formato estructurado que describa el problema y la solución, y se les da un nombre, podrían llamarse “Patrones”.

Un patrón es un par problema/solución con nombre, que se puede aplicar en contextos; con consejos acerca de cómo aplicarlo en situaciones y discusiones sobre sus compromisos.

Un patrón es una descripción de un problema bien conocido que suele incluir:

- Descripción.
- Escenario de Uso.
- Solución concreta.
- Las consecuencias de utilizar este patrón.
- Ejemplos de implementación.
- Lista de patrones relacionados.

1.8.2 Categorías de patrones

Según la escala o nivel de abstracción:

- *Patrones de arquitectura*: Aquéllos que expresan un esquema organizativo estructural fundamental para sistemas de software.

- *Idiomas:* Patrones de bajos niveles específicos para un lenguaje de programación o entorno concreto.

También es importante definir el concepto de Antipatrón de Diseño (5), que con forma semejante a la de un patrón, intenta prevenir contra errores comunes de diseño en el software. La idea de los Antipatrones es dar a conocer los problemas que acarrean ciertos diseños muy frecuentes, para intentar evitar que diferentes sistemas acaben una y otra vez en el mismo callejón sin salida, por haber cometido los mismos errores.

1.8.3.2 Patrones Estructurales

- **Adapter (Adaptador)**: Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.
- **Bridge (Puente)**: Desacopla una abstracción de su implementación.
- **Composite (Objeto compuesto)**: Permite tratar objetos compuestos como si de uno simple se tratase.
- **Decorator (Envoltorio)**: Añade funcionalidad a una clase dinámicamente.
- **Facade (Fachada)**: Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

- **Flyweight (Peso ligero)**: Reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.
- **Proxy**: (**Apoderado**) Mantiene un representante de un objeto.

Conclusiones parciales

A lo largo de este capítulo se han expuesto algunas características sobre las medias, para comprender y poder establecer una arquitectura que sustente las necesidades definidas para los productos del polo. Se efectuó un estudio de los principales estilos arquitectónicos, las características y definiciones de las diferentes arquitecturas presentes en cada uno de estos para lograr una selección adecuada para los productos del Departamento de Señales Digitales. Se realizó un análisis detallado de los diferentes patrones de diseño para una posterior selección. Se puede concluir que todo el estudio realizado en este capítulo es de vital importancia para cualquier sistema que se desee implementar, ya que a raíz de este estudio se crearán las bases para las arquitecturas que se pondrán en práctica en las futuras aplicaciones del Departamento de Señales Digitales.

2

CAPÍTULO

ATRIBUTOS DE CALIDAD Y ESCENARIOS ARQUITECTÓNICOS.

CAPÍTULO 2. ATRIBUTOS DE CALIDAD Y ESCENARIOS ARQUITECTÓNICOS.

2.1 Introducción

A partir de la búsqueda y el análisis de la información correspondiente a la calidad de un software, profundizando en los aspectos que definen la calidad y características que el sistema debe soportar, se hace necesario determinar los atributos de calidad así como especificar el/los escenarios arquitectónicos que lo describen.

2.2 Calidad del software

En la actualidad el desarrollo de software a nivel mundial ha alcanzado grandes niveles, es por ello que se le ha concedido una gran importancia a la calidad durante todo el proceso desarrollo de los mismos. La calidad de software se ha tomado muy en serio por las grandes conpañías de desarrollo de software, ya que permite competir con mayores posibilidades de éxito en el mercado internacional. Ella permite satisfacer las necesidades de los clientes, cumpliendo con los costos establecidos y en el tiempo que se ajustó con los clientes. Existen diversos criterios, uno de ellos es el de Pressman(2002 el cual plantea: es la concordancia con los requisitos funcionales y de

rendimiento establecidos con los estándares desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado de forma profesional. Por otro lado, la ISO/IEC (Organización Internacional de Estándares) la define como: la totalidad de rasgos y atributos de un producto de software que le apoyan en su capacidad de satisfacer sus necesidades explícitas o implícitas (ISO/IEC 9126, 19998) (30). Se llega a la conclusión de que existe la gran necesidad de que un software debe satisfacer los requisitos dados por un usuario.

Los requisitos no funcionales de un software influyen de gran manera en la calidad del mismo. Estos tienen gran impacto en el desarrollo y mantenimiento de un sistema, entre las más importantes se pueden destacar: la modificabilidad, eficiencia, mantenibilidad, interoperabilidad, confiabilidad, reusabilidad y facilidad de ejecución de pruebas. Estas características pueden ser identificadas independientemente del comportamiento del sistema, por lo que debe hacerse referencia a atributos de calidad en lugar de requisitos no funcionales.

2.3 Atributos de calidad

Los atributos de calidad de un sistema proveen una base en la toma de decisiones objetivas sobre acuerdos de diseño a la hora de realizar un producto software. El objetivo es lograr evaluar cuantitativamente y llegar a acuerdos entre múltiples atributos de calidad para alcanzar un mejor sistema de forma global. Tales atributos son requisitos no funcionales del sistema que hacen referencia a requisitos adicionales de este, los cuales serán diferentes de los requisitos funcionales. A modo de conclusión se puede mencionar que estos atributos definen las propiedades de un servicio que presta determinado sistema a los usuarios.

Bass(1998) establece una clasificación de atributos en general de dos categorías(30):

- Observables Vía Ejecución: que serán los atributos que se determinan del comportamiento del sistema en tiempo de ejecución.

Atributo de Calidad	Descripción
Disponibilidad *(Availability)*	Es la medida de disponibilidad del sistema para el uso (Barbacci et al., 1995).
Confidencialidad *(Confidentiality)*	Es la ausencia de acceso no autorizado a la información (Barbacci et al., 1995).
Funcionalidad *(Functionality)*	Habilidad del sistema para realizar el trabajo para el cual fue concebido (Kazman et al., 2001).
Desempeño *(Performance)*	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. (IEEE 610.12). Según Smith (1993), el desempeño de un sistema se refiere a aspectos temporales del comportamiento del mismo. Se refiere a capacidad de respuesta, ya sea el tiempo requerido para responder a aspectos específicos o el número de eventos procesados en un intervalo de tiempo. Según Bass et al. (1998), se refiere además a la cantidad de comunicación e interacción existente entre los componentes del sistema.
Confiabilidad *(Reliability)*	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo (Barbacci et al., 1995).
Seguridad externa *(Safety)*	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información (Barbacci et al., 1995).
Seguridad interna *(Security)*	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos (Kazman et al., 2001).

Tabla 1 Atributos en tiempo de ejecución.

- No observables Vía Ejecución: son aquellos atributos que se establecen durante el desarrollo del sistema.

Atributo de Calidad	Descripción
Configurabilidad *(Configurability)*	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema (Bosch et al., 1999).
Integrabilidad *(Integrability)*	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados. (Bass et al. 1998)
Integridad *(Integrity)*	Es la ausencia de alteraciones inapropiadas de la información (Barbacci et al., 1995).
Interoperabilidad *(Interoperability)*	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de *integrabilidad* (Bass et al. 1998)
Modificabilidad *(Modifiability)*	Es la habilidad de realizar cambios futuros al sistema. (Bosch et al. 1999).
Mantenibilidad *(Maintainability)*	Es la capacidad de someter a un sistema a reparaciones y evolución (Barbacci et al., 1995). Capacidad de modificar el sistema de manera rápida y a bajo costo (Bosch et al. 1999).
Portabilidad *(Portability)*	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos (Kazman et al., 2001).
Reusabilidad *(Reusability)*	Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones (Bass et al. 1998).
Escalabilidad *(Scalability)*	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental (Pressman, 2002).
Capacidad de Prueba *(Testability)*	Es la medida de la facilidad con la que el software, al ser sometido a una serie de pruebas, puede demostrar sus fallas. Es la probabilidad de que, asumiendo que tiene al menos una falla, el software fallará en su próxima ejecución de prueba (Bass et al. 1998).

Tabla 2 Atributos que se establecen durante el desarrollo del sistema.

2.4 Modelos de calidad

Para medir el nivel de complejidad de calidad de un sistema software, es necesario establecer algunos modelos, ya que la organización y descomposición de los atributos de calidad ha permitido el establecimiento de estos, los cuales serán específicos para la evaluación de la calidad en las arquitecturas.

Los modelos más importantes propuestos desde los años 70 y hasta la actualidad son:

- McCall (1997)
- Dromey (1996)
- FURPS (1987)

- ISO/IEC 9126 (1991)
- ISO/IEC 9126 : Adaptado para arquitecturas de software (2003)

2.4.1 Modelo de calidad de McCall

En el modelo de McCall,(30) los factores de calidad se concentran fundamentalmente en las características operativas, capacidad de cambios y adaptabilidad a nuevos entornos.

Factor	Criterio
Correctitud	✓ Rastreabilidad ✓ Completitud ✓ Consistencia
Confiabilidad	✓ Consistencia ✓ Exactitud ✓ Tolerancia a fallas
Eficiencia	✓ Eficiencia de ejecución ✓ Eficiencia de almacenamiento
Integridad	✓ Control de acceso ✓ Auditoría de acceso
Usabilidad	✓ Operabilidad ✓ Entrenamiento ✓ Comunicación
Mantenibilidad	✓ Simplicidad ✓ Concreción
Capacidad de Prueba	✓ Simplicidad ✓ Instrumentación ✓ Auto-descriptividad ✓ Modularidad
Flexibilidad	✓ Auto-descriptividad ✓ Capacidad de expansión ✓ Generalidad ✓ Modularidad
Portabilidad	✓ Auto-descriptividad ✓ Independencia del sistema ✓ Independencia de máquina
Reusabilidad	✓ Auto-descriptividad ✓ Generalidad ✓ Modularidad ✓ Independencia del sistema ✓ Independencia de máquina
Interoperabilidad	✓ Modularidad ✓ Similitud de comunicación ✓ Similitud de datos.

Tabla 3 Modelo de McCall, los factores de calidad y sus criterios asociados.

2.4.2 Modelo de calidad de Dromey

Dromey propuso un marco de referencia para la construcción de modelos de calidad (30). Sugiere el uso de cuatro categorías que implican propiedades de calidad: correctitud, internas, contextuales y descriptivas.

Propiedades del producto	Atributos de Calidad
Correctitud	✓ Funcionalidad ✓ Confiabilidad
Internas	✓ Mantenibilidad ✓ Eficiencia ✓ Confiabilidad
Contextuales	✓ Mantenibilidad ✓ Reusabilidad ✓ Portabilidad ✓ Confiabilidad
Descriptivas	✓ Mantenibilidad ✓ Reusabilidad ✓ Portabilidad ✓ Usabilidad

Tabla 4 Relación que establece Dromey entre las propiedades de calidad de un producto y los atributos de calidad.

2.4.3 Modelo de calidad de FURPS

En este modelo se desarrollan un conjunto de factores de calidad bajo las siglas de FURPS (30): funcionalidad (Functionality), usabilidad (usability), confiabilidad (Reliability), desempeño (Performance) y capacidad de soporte (Supportability).

Factor de Calidad	Atributos
Funcionalidad	✓ Características y capacidades del programa ✓ Generalidad de las funciones ✓ Seguridad del sistema
Facilidad de uso	✓ Factores humanos ✓ Factores estéticos ✓ Consistencia de la interfaz ✓ Documentación
Confiabilidad	✓ Frecuencia y severidad de las fallas ✓ Exactitud de las salidas ✓ Tiempo medio de fallos ✓ Capacidad de recuperación ante fallas ✓ Capacidad de predicción
Rendimiento	✓ Velocidad del procesamiento ✓ Tiempo de respuesta ✓ Consumo de recursos ✓ Rendimiento efectivo total ✓ Eficacia
Capacidad de Soporte	✓ Extensibilidad ✓ Adaptabilidad ✓ Capacidad de pruebas ✓ Capacidad de configuración ✓ Compatibilidad ✓ Requisitos de instalación

Tabla 5 Clasificación de los atributos de calidad que incluyen el modelo de FURPS , en conjunto con las características asociadas a cada uno de ellos.

2.4.4 Modelo ISO/IEC 9126

El estándar ISO/IEC 9126 fue desarrollado en un intento de identificar los atributos claves de calidad para un producto de software (30). Este estándar es una simplificación del modelo de McCall e identifica 6 características básicas de calidad que puede estar presentes en cualquier producto software.

Característica	Subcaracterística
Funcionalidad	✓ Adecuación ✓ Exactitud ✓ Interoperabilidad ✓ Seguridad
Confiabilidad	✓ Madurez ✓ Tolerancia a fallas ✓ Recuperabilidad
Usabilidad	✓ Entendibilidad ✓ Capacidad de aprendizaje ✓ Operabilidad
Eficiencia	✓ Comportamiento en tiempo ✓ Comportamiento de recursos
Mantenibilidad	✓ Analizabilidad ✓ Modificabilidad ✓ Estabilidad ✓ Capacidad de pruebas
Portabilidad	✓ Adaptabilidad ✓ Instalabilidad ✓ Reemplazabilidad

Tabla 6 Características con subcaracterísticas de modelo ISO/IEC 9126.

2.4.4.1 Modelo ISO/IEC 9126 adaptado para arquitecturas de software.

Se propone una adaptación del modelo ISO/IEC 9126 para la evaluación de arquitecturas de software. El modelo se basa en atributos de calidad que se relacionan directamente con la arquitectura: funcionalidad, confiabilidad, eficiencia, mantenibilidad y portabilidad.

Característica	Subcaracterística	Elementos de tipo arquitectónico
Funcionalidad	Adecuación	Refinamiento de los diagramas de secuencia
	Exactitud	Identificación de los componentes con las funciones responsables de los cálculos
	Interoperabilidad	Identificación de conectores de comunicación con sistemas externos
	Seguridad	Mecanismos o dispositivos que realizan explícitamente la tarea
Confiabilidad	Tolerancia a fallas	Existencia de mecanismos o dispositivos de software para manejar excepciones
	Recuperabilidad	Existencia de mecanismos o dispositivos de software para reestablecer el nivel de desempeño y recuperar datos
Eficiencia	Desempeño	Componentes involucrados en un flujo de ejecución para una funcionalidad
	Utilización de recursos	Relación de los componentes en términos de espacio y tiempo
Mantenibilidad	Acoplamiento	Interacciones entre componentes
	Modularidad	Número de componentes que dependen de un componente
Portabilidad	Adaptabilidad	Presencia de mecanismos de adaptación
	Instalabilidad	Presencia de mecanismos de instalación
	Coexistencia	Presencia de mecanismos que faciliten la coexistencia
	Reemplazabilidad	Lista de componentes reemplazables para cada componente

Tabla 7: Modelo ISO/IEC adaptadas a arquitecturas de software con elementos arquitectónicos asociados.

2.5 Comparación entre modelos de calidad.

Para realizar la propuesta de los atributos de calidad se tendrán en cuenta que estos coincidan con un mínimo de 3 ocasiones (60 %) en los 5 modelos de calidad estudiados.

Atributos	Dromey	McCall	FURPS	ISO9126	ISO Arquitectura
Disponibilidad					
Funcionalidad	X		X	X	X
Desempeño					
Confiabilidad	X	X	X	X	X
Seguridad Externa					
Seguridad Interna					
Configurabilidad					
Integralidad					
Integridad		X			
Interoperabilidad		X			
Modificabilidad					
Mantenibilidad	X	X		X	X
Portabilidad	X	X		X	X
Reusabilidad	X	X			
Escalabilidad					
Capacidad de Prueba		X			
Confidencialidad					
Correctitud		X			
Eficiencia	X	X		X	X
Usabilidad	X	X		X	
Flexibilidad					
Facilidad de uso			X		
Rendimiento			X		
Capacidad de soporte			X		

Tabla 8: Comparación de Atributos entre Modelos de Calidad:

2.5.1 Puntos Comunes

- Funcionalidad
- Confiabilidad
- Mantenibilidad
- Portabilidad
- Eficiencia
- Usabilidad

Los atributos de calidad *Eficiencia* y *Usabilidad* aunque no son parte de los atributos generales que se muestran en las tablas 1 y 2 , estarán incluídos en la propuesta de atributos para los productos del Departamento de Señales Digitales, ya que en su gran mayoría estos modelos de calidad los indican como otros evaluadores de las propiedades no funcionales de un sistema.

2.6 Comparación entre Arquitecturas.

A partir del estudio de las arquitecturas de software de los productos del Departamento de Señales Digitales, se determinaron los distintos atributos de calidad que satisfacen dichas arquitecturas.

Productos del Departamento	Arquitectura empleada	Atributos relacionados
Captura y Catalogación	Arquitectura centralizada	Integrabilidad Escalabilidad Modificabilidad
Video Web	Modelo_Vista_Controlador	Escalabilidad Desempeño Modificabilidad
Plataforma de radio y televisión	Modelo_Vista_Controlador	Escalabilidad Modificabilidad Desempeño
Primicia	Modelo_Vista_Controlador	Modificabilidad Escalabilidad Desempeño
Sistema de Gestión de Procesos para la Dirección de la Televisión Universitaria.	Modelo_Vista_Controlador	Escalabilidad Modificabilidad Desempeño

Tabla 9: Atributos de calidad propuestos por las arquitecturas de software empleadas en los productos del Departamento de Señales Digitales.

Una vez analizados los atributos de calidad de las diferentes arquitecturas de software se realizó una comparación para determinar el grado de coincidencia de los mismos y seleccionar los que formarán parte de la propuesta.

Atributo de Calidad	Arquitectura Centralizada	Arquitectura Modelo_Vista_Controlador
Desempeño		X
Integralidad	X	
Modificabilidad	X	X
Escalabilidad	X	X

Tabla 10: Comparación entre Atributos de calidad relacionados con las arquitecturas de software empleadas en los productos del Departamento de Señales Digitales.

2.6 .1 Puntos comunes

- Modificabilidad
- Escalabilidad

Se concluye que los atributos de calidad antes mencionados coinciden en las distintas arquitecturas empleadas para el desarrollo de los productos del Departamento de Señales Digitales, por lo que formarán parte de la propuesta.

2.7 Propuesta final de atributos

- Funcionalidad
- Confiabilidad
- Mantenibilidad
- Portabilidad
- Modificabilidad
- Escalabilidad
- Eficiencia
- Usabilidad

2.7.1 Funcionalidad

Se refiere a la habilidad del sistema de realizar el trabajo para el cual fue concebido. Este factor de calidad tiene características asociadas como las que se refieren a las capacidades de un programa, la generalidad de las funciones y la seguridad del sistema. Tiene subcaracterísticas asociadas como la adecuación, quien se refiere al

refinamiento de los diagramas de secuencia. Presenta exactitud, que hace alusión a la identificación de componentes con las funciones respuesta de los cálculos. Interoperabilidad, quien se encarga de identificar los conectores de comunicación con sistemas externos y seguridad, que es el mecanismo que realiza explícitamente la tarea.

2.7.2 Confiabilidad

Hace referencia a consistencia. Tiene algunas subcaracterísticas: una de ellas es la exactitud, quien representa la precisión de los cálculos y el control; tolerancia a fallos, estos son los atributos del software que proporcionan una estructura de módulos altamente independientes; correctitud; recuperabilidad, la que constituye la existencia de mecanismos o dispositivos de software para restablecer el nivel de desempeño y recuperar datos; exactitud de las salidas; tiempo medio de fallos; capacidad de recuperación ante fallas y capacidad de predicción entre otras.

2.7.3 Mantenibilidad

Es la capacidad de someter a un sistema a reparaciones y evolución. Se identifica como las propiedades internas de un producto. Presenta simplicidad, que se refiere a los atributos del software que posibilitan la implementación de funciones de la forma más comprensible posible. Concreción, quien se caracteriza por presentar atributos del software que posibilitan la implementación de una función con la menor cantidad de códigos posibles. Los atributos del software proporcionan explicaciones sobre la implementación de las funciones y proporcionan uniformidad en las técnicas y notaciones de diseño e implementación.

2.7.4 Portabilidad

Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos. Se caracteriza por atributos del software que proporcionan explicaciones sobre la implementación de las funciones; atributos del software que proporcionan módulos altamente independientes; características del software que determinan su independencia del entorno operativo y del hardware.

2.7.5 Modificabilidad

Modificabilidad hace refencia a la habilidad de hacer cambios futuros al sistema. Es un atributo de calidad que se asocia a los diferentes patrones arquitectónicos, principalmente al Modelo-Vista-Controlador.

2.7.6 Escalabilidad

Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental. Es un atributo que se asocia a la arquitectura centralizada y al estilo Modelo-Vista-Controlador.

2.7.7 Eficiencia

Se caracteriza por un conjunto de atributos con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas. Eficiencia hace alusión al comportamiento en el tiempo del software y al comportamiento de recursos, además de atributos que miniminizan el espacio de almacenamiento necesario.

2.7.8 Usabilidad

Facilidad con la cual las personas que interactúan con la aplicación en forma efectiva y sobre el sistema, provee soporte al usuario en este sentido. El sistema debe maximizar la información provista al usuario que tiende a aumentar su confianza en que no está cometiendo errores, y poder cancelar operaciones que no han finalizado. Una vez terminada una transacción esta no podrá ser reversada si implica un cambio de custodia de valores.

2.8 Escenarios

Se utilizan para resolver el problema de especificar los atributos de calidad y vincularlos con cada aspecto común de un sistema. La característica de un atributo estará dada por uno o varios escenarios.

2.8.1 Ejemplo de escenarios para los atributos de calidad propuestos.

Escenario	Interoperabilidad
Objetivo del negocio	Brindar servicios de audio y video a través de la Web
Atributo	Funcionalidad
Atributo de calidad relevante	Identificación de conectores de comunicación con sistemas externos
Fuente	Usuario autorizado
Fuente de estímulo	Usar protocolos para comunicarse
Ambiente	Comunicación entre el cliente y el servidor Web y entre el cliente y el servidor Streaming.
Respuesta	Se necesitan protocolos idóneos para la comunicación entre el cliente y estos servidores
Medida de Respuesta	Se aconseja usar el protocolo http entre el cliente y el servidor Web y rtsp entre el

Tabla 10: Ejemplo de escenario del atributo de calidad "*Funcionalidad"* respecto a los productos del Departamento de Señales Digitales

Escenario	Generar e imprimir reportes
Objetivo del negocio	El subsistema deberá ser capaz de generar e imprimir reportes
Atributo	Funcionalidad
Atributo de calidad relevante	Habilidad del sistema para realizar el trabajo para el cual fue concebido
Fuente	Usuario autorizado
Fuente de estímulo	El subsistema debe generar e imprimir reportes
Ambiente	Los datos pueden crecer en el servidor de medias y un poco menos en el servidor de base de datos
Respuesta	El sistema debe responder de manera eficiente y rápida
Medida de Respuesta	En un tiempo relativamente a las peticiones del usuario, el tiempo de respuesta debe ser menos de 5 segundos

Tabla 11: Ejemplo de escenario del atributo de calidad "*Funcionalidad"* respecto a los productos del Departamento de Señales Digitales

Escenario	Generar e imprimir reportes
Objetivo del negocio	El subsistema deberá ser capaz de generar e imprimir reportes
Atributo	Modificabilidad
Atributo de calidad relevante	Habilidad de realizar cambios futuros
Fuente	Usuario autorizado
Fuente de estímulo	Asignación de un equipamiento informático
Ambiente	Interfaz del sistema
Respuesta	Se modifica un plan de producción
Medida de Respuesta	El diseño debe ser altamente configurable

Tabla 12: Ejemplo de escenario del atributo de calidad "*Modificabilidad"* respecto a los productos del Departamento de Señales Digitales.

Escenario	Ejecución del sistema
Objetivo del negocio	Brindar servicios de audio y video a través de la Web
Atributo	Mantenibilidad
Atributo de calidad relevante	Número de componentes que dependen de otros componentes
Fuente	Usuario autorizado
Fuente de estímulo	Aumento del tiempo de respuesta en la ejecución del sistema
Ambiente	Lado del cliente
Respuesta	El tiempo de ejecución puede variar en dependencia de la precarga de archivos
Medida de Respuesta	El subsistema de administración debe posibilitar la visualización de los recursos multimedia almacenados

Tabla 14 : Ejemplo de escenario del atributo de calidad "*Mantenibilidad"* respecto a los productos del Departamento de Señales Digitales

Escenario	Generar e imprimir reportes
Objetivo del negocio	El subsistema deberá ser capaz de generar e imprimir reportes
Atributo	Confiabilidad
Atributo de calidad relevante	Habilidad del sistema a mantenerse en funcionamiento a lo largo de todo el tiempo
Fuente	Usuario autorizado
Fuente de estímulo	El subsistema de administración debe generar e imprimir reportes
Ambiente	La Base de Datos
Respuesta	El sistema debe resistir una alta concurrencia a lo largo de todo el período de operación del software
Medida de Respuesta	S aconsejable hacer réplicas de la base de datos en cada una de las PC clientes para que el sistema responda con mayor rapidez

Tabla 15: Ejemplo de escenario del atributo de calidad "*Confiabilidad"* respecto a los productos del Departamento de Señales Digitales

Escenario	Generar e imprimir reportes
Objetivo del negocio	El subsistema deberá ser capaz de generar e imprimir reportes
Atributo	Confiabilidad
Atributo de calidad relevante	Habilidad del sistema a mantenerse en funcionamiento a lo largo de todo el tiempo
Fuente	Usuario autorizado
Fuente de estímulo	El subsistema de administración debe generar e imprimir reportes
Ambiente	La Base de Datos
Respuesta	El sistema debe resistir una alta concurrencia a lo largo de todo el período de operación del software
Medida de Respuesta	S aconsejable hacer réplicas de la base de datos en cada una de las PC clientes para que el sistema responda con mayor rapidez

Tabla 16: Ejemplo de escenario del atributo de calidad "*Escalabilidad*" respecto a los productos del Departamento de Señales Digitales

Escenario	Operación en varios sistemas operativos
Objetivo del negocio	Visualización de recursos multimedia almacenados
Atributo	Portabilidad
Atributo de calidad relevante	El sistema puede ser ejecutado en diferentes ambientes
Fuente	Usuario autorizado
Fuente de estímulo	Implementar el sistema sobre el lenguaje de programación php
Ambiente	Diferentes plataformas
Respuesta	Resulta provechoso a la hora de una migración de sistemas operativos
Medida de Respuesta	Es aconsejable que el sistema opere en varios sistemas operativos sin necesidad de modificar el código que se encuentra

Tabla 17: Ejemplo de escenario del atributo de calidad "*Portabilidad"* respecto a los productos del Departamento de Señales Digitales.

Escenario	Diseño
Objetivo del negocio	Brindar servicios de audio y video a través de la Web
Atributo	Escalabilidad
Atributo de calidad relevante	Ampliar diseño arquitectónico
Fuente	Usuario autorizado
Fuente de estímulo	Implementar arquitectura para componentes
Ambiente	Web
Respuesta	Arquitectura que permita la fácil integración o desintegración de componentes
Medida de Respuesta	Se recomienda implementar arquitectura flexible

Tabla 18: Ejemplo de escenario del atributo de calidad "*Escalabilidad"* respecto a los productos del Departamento de Señales Digitales

Escenario	Servicios de audio y video
Objetivo del negocio	Brindar servicios de audio y video a través de la Web
Atributo	Eficiencia
Atributo de calidad relevante	Comportamiento en el tiempo
Fuente	Usuario autorizado
Fuente de estímulo	Brindar servicios de audio y video
Ambiente	Web
Respuesta	El sistema debe responder en un tiempo relativamente rápido a las peticiones de los usuarios
Medida de Respuesta	Es aconsejable que en un tiempo de

Tabla 19: Ejemplo de escenario del atributo de calidad "*Eficiencia"* respecto a los productos del Departamento de Señales Digitales

Escenario	Servicios de audio y video
Objetivo del negocio	Brindar servicios de audio y video a través de la Web
Atributo	Confiabilidad
Atributo de calidad relevante	Seguridad del sistema
Fuente	Usuario autorizado/Usuario no autorizado
Fuente de estímulo	Acceder a la aplicación
Ambiente	Web
Respuesta	Solo tendrán acceso al sistema usuarios de la plataforma Web
Medida de Respuesta	Se aconseja el uso de contraseñas encriptadas usando el algoritmo MD5

Tabla 20: Ejemplo de escenario del atributo de calidad "*Confiabilidad"* respecto a los productos del Departamento de Señales Digitales.

Escenario	Generar e imprimir reportes
Objetivo del negocio	El subsistema deberá ser capaz de generar e imprimir reportes
Atributo	Mantenibilidad
Atributo de calidad relevante	Capacidad de someter a un sistema a reparaciones y evolución
Fuente	Usuario autorizado
Fuente de estímulo	Gestionar plan de programación
Ambiente	El sistema
Respuesta	Se podrá gestionar un nuevo plan de programación , modificar uno ya existente o eliminar alguno que se desee
Medida de Respuesta	Se necesitan esas posibilidades para un

Tabla 21: Ejemplo de escenario del atributo de calidad "*Mantenibilidad"* respecto a los productos del Departamento de Señales Digitales.

Escenario	Generar e imprimir reportes
Objetivo del negocio	El subsistema de adminstrción sserá capaz de generar e imprimir reportes
Atributo	Eficiencia
Atributo de calidad relevante	Comportamiento en el tiempo
Fuente	Usuario autorizado
Fuente de estímulo	El subsistema debe generar e imprimir reportes
Ambiente	Los datos pueden crecer en el servidor de medias y un poco menos en el servidor de base de datos
Respuesta	El sistema debe responder de manera eficiente y rápida
Medida de Respuesta	El tiempo de respuesta debe ser menor que 5 segundos

Tabla 22: Ejemplo de escenario del atributo de calidad "*Eficiencia"* respecto a los productos del Departamento de Señales Digitales.

2.9 Conclusiones Parciales

A lo largo de este capítulo se mencionaron algunas características sobre la importancia que tiene la calidad en los proyectos de desarrollo de software

posibilitando la selección de los atributos de calidad que sustentarán las futuras aplicaciones que se construyan en el Departamento de Señales Digitales. Se establecieron escenarios como mecanismos para describir y representar los atributos de calidad propuestos. Se concluye que los atributos: Eficiencia y Usabilidad no son parte de los atributos generales, al formar parte Modelos de Calidad y tener un alto grado de coincidencia en los mismos, estos integran la propuesta de solución.

3

CAPÍTULO

TECNOLOGÍAS DE DESARROLLO Y TELEMÁTICAS.

CAPÍTULO 3:TECNOLOGÍAS DE DESARROLLO Y TELEMÁTICAS.

A partir del análisis realizado sobre las diferentes tecnologías de servicios que serán de gran ayuda para desarrollar un software. Se proponen la/las arquitecturas de software idóneas para los productos del Departamento de Señales Digitales. Se definen además los protocolos con que se comunicaránlos futuros productos que se implementen en el Departamento de Señales Digitales. Se especifican además las herramientas que se utilizarán en el proceso de desarrollo de las aplicaciones que se quieran construir.

3.1 Metodologías de Desarrollo de Software.

La complejidad de los proyectos de software hoy en día, el constante cambio de requisitos y la falta de documentación, provocan que los proyectos se retrasen en tiempo y se incrementen en costo. La solución a esta problemática es implantar una arquitectura de desarrollo que permita darle seguimiento a los proyectos desde su etapa de requisitos, hasta su implantación.

Todo proceso de software es riesgoso y difícil de controlar, si no se lleva a cabo una metodología de desarrollo de software, no se obtienen los resultados que se necesitan y el cliente no queda satisfecho.

3.1.1 EXTREME PROGRAMING (XP).

Extreme Programing surge como una nueva manera de encarar proyectos de software, proponiendo una metodología basada esencialmente en la simplicidad y

agilidad. Tiene características totalmente diferentes de la planificación tradicional de cualquier proyecto de software. Hace referencia a una programación extrema que plantea que es imposible prever todo antes de comenzar a programar. XP es una de las llamadas metodologías ágiles de desarrollo de software más exitosos de los tiempos recientes. Se caracteriza porque el cliente o el usuario se convierten en miembro del equipo de desarrollo. Introduce funcionalidades cuando sean necesarias, no antes, o sea, se irán añadiendo las funcionalidades continuamente. Plantea que el cliente tiene el derecho de saber el estado del proyecto, así como decidir lo que se implementa. A su vez el desarrollador decidirá cómo se van a implementar los procesos además de implementar el sistema con la mayor calidad posible. Lo principal en este tipo de tecnología es la comunicación que va a haber entre usuarios y desarrolladores, la simplicidad y la retroalimentación concreta del equipo de desarrollo, del cliente y de los usuarios finales.

XP se basa en los siguientes principios (48):

- Satisfacer al cliente a través de entregas continuas y tempranas es la mayor prioridad.

- Los cambios a los requisitos son bienvenidos, aún en fases tardías de desarrollo.

- Entregar frecuentemente un software que funciona, desde un par de semanas a un par de meses, prefiriendo los periodos más cortos.

- Desarrolladores, gerentes y clientes deben trabajar juntos diariamente, a lo largo del proyecto.

- Construir proyectos alrededor de personas motivadas, dándoles el entorno y soporte que necesitan y confiando en que realizarán el trabajo.

- El método más eficiente y efectivo de transmitir información entre un equipo de desarrolladores es la conversación frontal (cara a cara).

- Tener un software que funcione es la medida primaria del progreso.

- El proceso ágil promueve el desarrollo sostenible. Los desarrolladores y usuarios deben ser capaces de mantener un ritmo de trabajo constante en forma permanente a lo largo del proyecto.

- La atención continua a la excelencia técnica y al buen diseño, mejoran la agilidad.

- A intervalos regulares el equipo debe reflexionar sobre como ser más efectivos y ajustar su comportamiento de acuerdo a ello.

3.1.2 RATIONAL UNIFIED PROCESS (RUP).

La metodología RUP define quién, cómo, cuándo y qué debe hacerse. Aporta diversas herramientas que son de gran uso para el desarrollo de una aplicación, ejemplo de ello son los casos de uso. Dicha metodología utiliza el paradigma orientado a objetos para su descripción e implementa las mejores prácticas de desarrollo de software. Es un marco que se utiliza para resolver necesidades específicas.

RUP está guiado por casos de uso, está centrado en la arquitectura y es iterativo e incremental. Es una de las metodologías más importantes, ya que en el transcurso de desarrollo de una aplicación exige el uso de diversos artefactos que proporcionan cierto grado de certificación en dicho proceso.

Las iteraciones tempranas de proyectos conducidos por el Proceso Unificado de Desarrollo (RUP) se enfocan fuertemente en la arquitectura del software. La puesta en práctica rápida de características se retrasa hasta que se ha establecido una arquitectura firme. La ventaja principal de esta metodología es que basa en las mejores prácticas que se han desarrollado y probado. Es una metodología de gran importancia, ya que permite documentar todo el proceso de desarrollo de un software. Si algún miembro del equipo de desarrolladores no puede seguir implementando, el otro que ocupe su lugar tendrá por donde documentarse y guiarse sobre lo que se ha realizado hasta el momento.

3.1.3 METODOLOGÍA PROPUESTA.

A partir del estudio detallado de las metodologías de desarrollo de software antes mencionadas, se selecciona para los productos del Departamento de Señales

Digitales "*RUP"*. La misma está dividida en 4 fases (Conceptualización, Elaboración, Construcción, Transición) y 9 flujos de desarrollo, posibilitando dividir el desarrollo del software en pequeños ciclos e iteraciones. Garantiza la elaboración de todas las fases de un producto de software orientado a objetos. Es una metodología de desarrollo de software, que emplea los distintos diagramas que propone UML, constituyendo la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas. La misma permite realizar un levantamiento exhaustivo de requisitos. Detecta defectos en etapas tempranas del proceso de desarrollo de software. Intenta reducir el número de cambios tanto como sea posible. Realiza el análisis y diseño tan completo como sea posible. Intenta anticiparse a futuras necesidades. El cliente interactúa con el equipo de desarrollo mediante reuniones y no es necesario que esté permanentemente presente en todo momento, a diferencia de la metodología XP que el cliente es parte del equipo de desarrollo. Se puede destacar como un elemento importante también que los equipos de desarrollo del Departamento de Señales Digitales están familiarizados con dicha metodología.

3.2 Lenguajes de Modelado.

Se necesita un lenguaje para visualizar, especificar, construir y documentar los artefactos que se vayan generando a lo largo de la construcción de un producto software.

3.2.1 Lenguaje de Modelado BPMN (Business Process Modeling Notation).

BPMN es un estándar de modelado de procesos de negocio, donde se presentan gráficamente las diferentes etapas del proceso del mismo. La notación ha sido diseñada específicamente para coordinar la secuencia de procesos y los mensajes que fluyen entre los diferentes procesos participantes. Dicha metodología ha sido desarrollada para proveer a los usuarios de una notación de uso libre. Esto beneficiará a los usuarios de la misma forma que UML benefició el mundo de la ingeniería de software (42). Está dirigido a gerentes, directores, dueños de empresas, ingenieros de procesos, analistas de negocios, analistas de sistemas, administradores de proyectos, responsables de calidad y todo aquel que necesita definir, documentar y hacer más eficientes sus procesos de negocio. El principal objetivo de BPMN es proveer una notación estándar que sea fácilmente comprensible por todos los accionistas de la

empresa. Los analistas de negocios crean y refinan los procesos, los desarrolladores técnicos son los responsables de la aplicación de los procesos y los gerentes de negocios supervisan y gestionan los procesosSirve como lenguaje común para salvar la brecha de comunicación que con frecuencia se produce entre el proceso de diseño de negocios y su aplicación.

3.2.2 LENGUAJE UNIFICADO DE MODELADO (UML).

UML aprovechó la experiencia de sus creadores, eliminó los componentes que resultaban de poca utilidad práctica añadiendo otros elementos. Es expresivo, claro y uniforme, aunque no garantiza el éxito de los proyectos, si mejora sustancialmente el desarrollo de los mismos al permitir una fuerte integración entre las herramientas, los procesos y los dominios. Se puede usar para modelar distintos tipos de sistemas: sistemas de software, de hardware, etc. Permite que se represente de manera semi-formal la estructura general del sistema, con la ventaja de que este mismo lenguaje puede ser usado en todas las etapas de desarrollo del sistema y su representación gráfica puede ser usada para comunicarse con los usuarios. Es independiente del lenguaje de programación y de las características de los proyectos, ya que ha sido diseñado para modelar cualquier tipo de proyectos. Este lenguaje permite la modelación de los componentes del proceso de desarrollo de aplicaciones que estén orientadas a objetos. Es un lenguaje gráfico con una sintaxis bien definida que pretende unificar las prácticas pasadas sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar (36). La representación en UML de un software estará dada por 4+1 Vista o modelos separados entre sí. Estas vistas son las siguientes:

- Vista de casos de uso.
- Vista lógica.
- Vista de implementación.
- Vista de procesos.
- Vista de despliegue.

3.2.3 LENGUAJE DE MODELADO SELECCIONADO.

Se selecciona UML como lenguaje de modelado para los futuros productos del Departamento de Señales Digitales ya que se enfoca en el diseño del software. Entre las características fundamentales por la que se escoge es porque toma un perfil orientado a objetos en el modelado de aplicaciones, mientras que BPMN toma un perfil orientado a procesos. Las extensiones de UML para el modelado de negocio aportan elementos muy importantes ya que proporcionan otras vistas de la arquitectura de negocio que son más difíciles de observar usando únicamente BPMN, por ejemplo, la visualización de las responsabilidades de los trabajadores del negocio, la manipulación de las entidades del negocio y la comprensión de los estados asociados a las entidades del negocio. Dentro de las principales características se puede mencionar que UML es el más empleado a nivel mundial, además de que los equipos de desarrollo del Departamento de Señales Digitales están familiarizados con dicho lenguaje.

3.3 Herramientas CASE.

Las herramientas CASE (Computer Aided Software Engineering) o Ingeniería de Software Asistida por Computadora en español, son herramientas de modelado usadas en el mundo de la informática que estarán consignadas al aumento de la productividad de software disminuyendo el coste de estos. Ayudan en gran medida en la realización del diseño de un producto, la detección de errores, cálculo de costes entre otras. Estas herramientas son muy usadas por los implementadores que buscan el logro de grandes resultados en la construcción de un software. Las herramientas CASE fueron creadas con el objetivo de cubrir las necesidades de los desarrolladores de sistemas para la automatización de los procesos, diseño y análisis de un proyecto software.

3.3.1 VISUAL PARADIGM 6.4

Visual Paradigm es muy sencillo de usar y a su vez fácil de instalar. Emplea los diagramas de UML para el modelado. Permite la fabricación de diversos tipos de diagramas completamente visuales. Dicha herramienta posibilita ver la aplicación desde diversos puntos de vista a través de la representación de diagramas, como pueden ser los de secuencia, casos de uso, actividad entre otros. Es una herramienta de modelado de gran alcance que apoya el ciclo de vida completo del software, análisis, codificación, diseño, prueba y despliegue. El lenguaje gráfico que usa permite

una mejor comunicación entre los miembros del equipo de realización de un software. Soporta múltiples usuarios trabajando en el mismo producto. Genera la documentación del proyecto automáticamente en formatos como Web o Pdf, además de permitir el control de versiones. Es una herramienta robusta y además portable. Es multiplataforma y utilizada en la construcción de grandes e importantes proyectos. Tiene gran alcance, es fácil de utilizar y apoya el ciclo de vida completo del software: análisis, diseño, codificación, prueba y despliegue. Se pueden diseñar diagramas UML, generar código de diagramas de la clase y viceversa y generar la documentación. Soporta todos los diagramas de la más reciente notación de UML. En adición al soporte de Modelado UML esta herramienta provee el modelado de procesos de negocios, además de un generador de mapeo de objetos-relacionales para los lenguajes de programación Java y PHP. Está disponible en varias ediciones, cada una destinada a unas necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Esta herramienta ayuda a los equipos de desarrollo de software a sobresalir todo el modelo de acumulación de trabajo así y desplegar el proceso de desarrollo de software, lo que permite maximizar y acelerar tanto las contribuciones individuales como las de equipo (43). La última versión de esta herramienta emplea una rápida respuesta con poca memoria utilizando moderadamente los tiempos del procesador, lo que le permite manejar grandes y complicadas estructuras de un proyecto en una forma muy eficiente. Los usuarios y proveedores de tecnología pueden integrar Visual Paradigm en cada uno de sus modelos para utilizarlos en sus soluciones con un mínimo esfuerzo. Esta herramienta proporciona una navegación intuitiva entre código y el modelo. Poderoso generador de documentación y reportes UML PDF/HTML/MS Word. Demanda en tiempo real, modelo incremental de viaje redondo y sincronización de código fuente. Diagramas de diseño automático sofisticado. Análisis de texto y soporte de tarjeta CRC entre otras.

3.3.2 RATIONAL ROSE.

Es una herramienta que cubre todo el ciclo de vida de un proyecto. Ofrece un lenguaje de modelado común que agiliza la creación del software. Es un entorno de modelado que agiliza la creación del software. El navegador UML que presenta dicha herramienta facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de casos de uso, lógica, de componentes y de despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción (39). Es una herramienta con plataforma independiente que ayuda a la comunicación entre los

miembros de un equipo, permite monitorear el tiempo de desarrollo y entender el entorno de los sistemas. Una de las grandes ventajas es que se basa en la notación estándar (UML), la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común. Los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto. Tiene un complemento de modelado Web que incluye funciones de visualización, modelado y herramientas para desarrollar aplicaciones Web. Tiene modelado en UML para diseñar bases de datos, el cual integra los requisitos de datos y aplicaciones mediante diseños lógicos y analíticos. Presenta la posibilidad de publicar en la Web modelos e informes para mejorar la comunicación entre los miembros del equipo.

3.3.3 Herramienta CASE propuesta.

Para desarrollar futuras aplicaciones en el Departamento de Señales Digitales, se ha seleccionado Visual Paradigm en su versión 6.4 ya que es poderosa y robusta. Es una herramienta compatible con ediciones anteriores. Presenta la ventaja de que se integra con varios IDE, mientras que Rational Rose Enterprise solamente con Borlan JBuilder y Microsoft Visual Studio y no en todas sus versiones. También Visual Paradigm es multiplataforma y Rational Rose no. Soporta las últimas versiones de UML generando un entorno de modelados visuales en el que se reúnen hoy todas las necesidades tanto de software y tecnología, así como de las necesidades de comunicación. Proporciona la generación de código y compatibilidad hasta con 10 lenguajes, ejemplo: C++ , CORBA IDL, PHP, XML, Schema, Ada, Python etc. Es un entorno superior de modelado visual. Entre las grandes características que también hacen que se proponga esta herramienta es que proporciona una mayor documentación de la base de datos y diagramas de mapeo de relación de objetos. Permite realizar ingeniería de ida y vuelta, ingeniería inversa (de código a modelo y de código a diagrama), posee un modelado colaborativo con Subversion y posee un editor de detalles de caso de uso.

3.4 Tecnologías de desarrollo de software.

El desarrollo de estas aplicaciones requiere del uso de algunas herramientas para desarrollarlas, dichas herramientas son aspectos importantes en la arquitectura de un proyecto.

3.4.1 LENGUAJE DE PROGRAMACIÓN PARA APLICACIONES WEB.

3.4.1.1 PHP

Sería idóneo utilizar PHP como lenguaje de programación puesto que es completamente orientado al desarrollo de aplicaciones Web. Se usa para la creación de contenido dinámico para este tipo de sitios. Ofrece soluciones simples y universales para las paginaciones dinámicas web de fácil programación. Entre sus características fundamentales se encuentra que es gratuito, ya que al tratarse de software libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre. Es de gran popularidad, pues existe una gran comunidad de desarrolladores y programadores que continuamente implementan mejoras en su código. Presenta una sencilla integración con múltiples bases de datos, esencial para una página web verdaderamente dinámica. Entre sus principales ventajas está el alto rendimiento que presenta, además de interfaces para una gran cantidad de sistemas de base de datos, diferentes Bibliotecas incorporadas para muchas tareas web habituales, bajo coste facilidad de aprendizaje y uso, portabilidad y acceso a código abierto. Actualmente se usa por su versatilidad de uso con diferentes sistemas operativos. Produce sensación al usuario de mayor rapidez y mayor usabilidad ya que es poco pesado. Mejora el soporte para la programación orientada a objetos. Presenta un mejor soporte para MySQL y un mejor rendimiento en general. Es multiplataforma y altamente usado por los programadores. Corre en cualquier plataforma usando el mismo código fuente. Puede interactuar con muchos motores de bases de datos. Su versión PHP 5.3.2 brinda mejoras significativas con respecto a versiones anteriores, con una orientación a objeto similar a Java. Utilizando PHP 5.3.2 las aplicaciones gozaran de nuevas capacidades, se obtendrá el beneficio de una mejor performance de ejecución (está comprobado experimentalmente que PHP5 corre un 25% más rápido que versiones anteriores) y el código estará muy bien acondicionado en cuanto a la compatibilidad con la versión que se asoma, PHP6.

3.4.2 LENGUAJES DE PROGRAMACIÓN PARA APLICACIONES DE ESCRITORIO.

3.4.2.1 JAVA

En el caso de las aplicaciones de *Escritorio*, para los sistemas del Departamento de Señales Digitales se proponen los lenguajes de programación Java y C++.

JAVA es un lenguaje de programación orientado a objetos. Se ha convertido en uno de los lenguajes más demandados por los desarrolladores. Está enfocado fundamentalmente en que el entorno de desarrollo de un producto software sea independiente de la plataforma en la que se ejecute. Ofrece un lenguaje altamente potente, similar al lenguaje C++ poniendo un poco más de énfasis en el tema de la seguridad. Fue creado con el objetivo de servir como una nueva manera de manejar la complejidad del software. Se usa en gran variedad de plataformas computacionales. La programación de JAVA permite el desarrollo de programas del rendimiento seguro y alto en las plataformas múltiples. Entre las ventajas que presenta este lenguaje de programación están las siguientes (44):

- Es una fuente abierta, así que los usuarios no tienen que luchar con los impuestos sobre patente pesados cada año
- Independiente de la plataforma
- Java realiza la colección de basura de las ayudas, así que la gerencia de memoria es automática
- Java asigna siempre objetos en el apilado
- Java abrazó el concepto de especificaciones de la excepción
- Usando JAVA se puede desarrollar aplicaciones webs dinámicas
- Permite crear programas modulares y códigos reutilizables.

3.4.2.2 C++

En cuanto al lenguaje de programación *C++* se puede decir que es un lenguaje combinado, estructurado, con un gran soporte para la programación orientada a objetos y procedural. Permite la reutilización de código de una manera productiva, los cuales pueden ser compilados en diversos sistemas operativos. Está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo, es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su

aprendizaje. Dentro de las principales características de este lenguaje de programación está el soporte para la programación orientada a objetos.

Las principales ventajas que presenta dicho lenguaje son (45):

- **Difusión:** al ser uno de los lenguajes más empleados en la actualidad, posee un gran número de usuarios y existe una gran cantidad de libros, cursos, páginas web etc. dedicados a él.
- **Versatilidad:** C++ es un lenguaje de propósito general, por lo que se puede emplear para resolver cualquier tipo de problema.
- **Portabilidad:** es uno de los lenguajes más rápidos en cuanto a ejecución.
- **Herramientas:** existe una gran catidad de compiladores, librerías, depuradores etc.

3.4.3 FRAMEWORKS DE DESARROLLO.

Conjunto de clases que cooperan y forman un diseño reutilizable para un tipo específico de software. Un framework ofrece una guía arquitectónica partiendo del diseño en clases abstractas y definiendo sus responsabilidades y sus colaboraciones. Un desarrollador personaliza el framework para una aplicación particular mediante herencia y composición de instancias de las clases del framework (33).

Es una infraestructura software que crea un entorno común para integrar aplicaciones e información dentro de un dominio dado(34). Es un marco de trabajo, el programa que se usa para el desarrollo de software en un lenguaje de programación determinado.

3.4.3.1 FRAMEWORK DE DESARROLLO PARA APLICACIONES WEB.

3.4.3.1.1 SYMFONY V-1.4

Para aplicaciones Web se empleará Symfony como framework de desarrollo. El mismo está implementado sobre una arquitectura de tipo Modelo_Vista_Controlador. Symfony es un framework diseñado para optimizar. Se caracteriza por separar la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Fácil de instalar y configurar en la mayoría de plataformas . Es independiente del sistema gestor de base de datos. Sencillo de usar y lo suficientemente flexible como para adaptarse a los

casos más complejos. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Está preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo. Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. Está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Sigue la mayoría de mejores prácticas y patrones de diseño para la web. Normalmente, todos los productos web comparten el mismo tipo de contenidos, como pueden ser una base de datos, archivos estáticos, archivos subidos al sitio web por parte de los usuarios o administradores, clases y librerías php, librerías externas, entre otros. Symfony proporciona una estructura en forma de árbol de archivos para organizar de forma lógica todos estos contenidos, además de ser consistente con la arquitectura MVC utilizada y con la agrupación proyecto / aplicación / módulo. Cada vez que se crea un nuevo proyecto, aplicación o módulo, se genera de forma automática la parte correspondiente de esa estructura. (40). Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows

3.4.3.2 FRAMEWORK PARA APLICACIONES DE ESCRITORIO.

3.4.3.2.1 HIBERNATE V - 3.7

Para las aplicaciones de Escritorio en caso de utilizar el lenguaje de programación Java se recomienda usar el framework Hibernate, el cual servirá de puente entre la capa de acceso a datos y la Base de Datos.

Sería más natural y sencillo trabajar directamente con objetos, pero es imposible con las BD relacionales, y las BD orientadas a objeto puesto que están todavía muy verdes. La mejor opción entonces es utilizar un motor de persistencia, que es el componente software encargado de traducir entre objetos y registros. Un motor de persistencia de código abierto es Hibernate, el cual permitirá guardar un objeto en la base de datos o borrarlo de forma automática.

Es una de herramienta de mapeo para el lenguaje Java que implementa numerosas funcionalidades. Permite solucionar el problema de la diferencia entre los datos orientados a objetos usados en la memoria de la computadora y el modelo relacional usado en la Base de Datos. Maneja el acceso a datos desde el punto de vista de la programación orientada a objetos. Utiliza el mecanismo de reflexión de Java, el cual se refiere al permiso que tiene un objeto en ejecución de examinarse a sí mismo. Entre sus ventajas se encuentra que presenta facilidad de programación y es idónea para aplicaciones transaccionales sin procesamiento masivo. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una Base de Datos ya existente. También tiene la funcionalidad de crear la Base de Datos a partir de la información disponible.

Presenta algunas características técnicas y son las que a continuación se mencionan (47):

- Modelo de programación natural. Hibernate es una capa de persistencia objeto/relacional que permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos.

- Gran escalabilidad. Hibernate es muy eficiente, tiene una arquitectura de caché de doble capa y podría ser usado en un cluster.

- Software libre. Está bajo licencia LGPL (Lesser GNU Public License).

- Persistencia transparente. Ofrece soporte para un amplio conjunto de las colecciones de Java, propiedades del estilo de persistencia de JavaBeans, etc. que abstraen al usuario.

- Mapeado flexible gracias a las asociaciones bidireccionales, la persistencia transitiva,colecciones de tipos básicos, etc. el mapeado resulta mucho más flexible.

- Facilidades en consultas. Debido en parte a que se realizan en un potente lenguaje de consultas orientado a objetos.

- Facilidades en metadatos. Soporta el formato del mapeado de XML, diseñado para ser editado a mano y el mapeado basado en anotaciones. Además de Validación basada en anotaciones.

3.4.3.2.2 QT 4.6

Qt es un framework para el desarrollo de aplicaciones multiplataforma. Dentro de sus características se encuentran:

- Compatibilidad multiplataforma con un sólo código fuente.
- Performance de C++.
- Disponibilidad del código fuente.
- Excelente documentación (Qt Assistant).
- Internacionalización de aplicaciones (Qt Linguist).
- Soporte para XML, conexión a bases de datos, SVG, OpenGL, programación para redes.
- Biblioteca de clases intuitiva
- Herramientas de desarrollo integrado
- Integración con Visual Studio
- Utilice los controles de ActiveX en aplicaciones Qt
- Integra Direct3D para el hardware de gráficos acelerados
- Soporte para aplicaciones 64-bits en la plataforma Mac con la API Cocoa.
- Librerías para analizar la carga de las aplicaciones haciendo benchmark de las mismas. La librería Qt posee numerosos módulos que permite trabajar con bases de datos, gráficos, imágenes, sonido, etc.

Las ventajas de usar este framework gráfico son extensas:

- Es usado en disímiles tecnologías como la telefonía celular (Symbian S606)1, aplicaciones de tipo SCADA, dispositivos multipropósito, y otras.

- Es multiplataforma. El mismo código que se escribe en GNU/Linux, puede ser ejecutado fácilmente en Windows XP o en Mac OSX.

- Por las amplias características con que cuenta, se hace muy fácil construir aplicaciones para distintos usos.

- Cuenta con un excelente diseñador gráfico de aplicaciones llamado Qt Designer, el cual hace más fácil todavía el trabajo del programador.

- Tiene un sistema integrado de internacionalización llamado Qt Linguist lo que permite portar las aplicaciones al lenguaje que se desee.

Las aplicaciones basadas en Qt tienen una buena respuesta y un buen uso de la memoria. El desarrollo con Qt es muy apropiado para proyectos software de larga escala, tanto comerciales como de libre distribución

El conjunto de herramientas tiene un enfoque más eficiente para facilitar la tarea de gestión de memoria para sus programadores: cuando un objeto es suprimido, todos los objetos dependientes se eliminan automáticamente. El enfoque está en no interferir con la libertad del programador para borrar manualmente cuando lo deseen. La biblioteca Qt está concebida como multiplataforma, permitiendo escribir código que se compilará y ejecutará en distintas plataformas, incluyendo Unix, Linux, FreeBSD o incluso Windows. Se pueden ejecutar aplicaciones de Qt así como aplicaciones de KDE sin tener que estar corriendo el mismo KDE.

Funciona en todas las principales plataformas y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros métodos para el manejo de ficheros, además de estructuras de datos tradicionales.

Qt es utilizada en KDE, un entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros. Utiliza el lenguaje de programación C++ de forma nativa,

adicionalmente puede ser utilizado en varios otros lenguajes de programación a través de *bindings*.

Incluye un framework de animación que ayuda a construir grandes animaciones, interfaces gráficas de alto rendimiento sin la molestia de la gestión de estructuras complejas, temporizadores, etc. También permite crear animaciones personalizadas y funciones de interpolación. Tiene una clase que define las funciones de la funcionalidad compartida por todas las animaciones, la clase QAbstractAnimation que es la base de todas las animaciones. QAbstractAnimation proporciona las funciones virtuales puras utilizadas por las subclases para seguir el progreso de la animación: la duración () y updateCurrentTime (). Las vistas gráficas no están omitidas, se puede animar QGraphicsWidgets y QGraphicsObject las cuales heredan de QGraphicsItem. Cuenta con efectos que pueden ser utilizados para alterar la apariencia de los elementos de la interfaz de usuario, un par de efectos estándar tales como visión borrosa, colorear, proporcionar sombras y es posible aplicar efectos personalizados Las animaciones son controladas usando las curvas fáciles y pueden ser agrupadas Esto permite que las animaciones sean de complejidad arbitraria. El API es fácil de entender con funciones tales como start (), stop (), pause (), y currentTime (). Qt cuenta actualmente con un sistema de triple licencia: GPL v2/v3 para el desarrollo de software de código abierto y software libre, la licencia de pago QPL para el desarrollo de aplicaciones comerciales, y a partir de la versión 4.5 una licencia gratuita pensada para aplicaciones comerciales, LGPL.

3.4.4 ENTORNO DE DESARROLLO INTEGRADO (IDE)

Un Entorno de Desarrollo Integrado es un programa informático que está compuesto por un conjunto de herramientas de programación. Es una plataforma donde se puede programar cómodamente.

3.4.4.1 NETBEANS 6.8

NetBeans es un proyecto de código abierto de gran éxito con una gran cantidad de usuarios. Es un producto libre, gratuito y sin restricciones de uso. Hace referencia a una plataforma para el desarrollo de aplicaciones de Escritorio. El soporte de Java Enterprise Edition es de importancia, en especial para los desarrolladores de JBuilder. Dado que cuenta con el mejor soporte a estándares industriales de la tecnología Java Ha hecho que el desarrollo de aplicaciones Java de tipo empresarial sea más rápido y

sencillo, su facilidad de uso, su cumplimiento de regulaciones, sus perfiles de rendimiento, además de su flexibilidad entre plataformas (37).

NetBeans le sirve a los programadores para escribir, compilar, depurar y ejecutar programas. Posee un navegador web integrado. Presenta el editor con más colores y un soporte más eficiente para XHTML marcando con diferentes colores las etiquetas de apertura y cierre. Permite mostrar el avance del proyecto y las tareas que se han terminado a través de un diagrama de Gantt. Tiene además un excelente completamiento de código, pistas de error y ventanas emergentes de documentación.

Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Además es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans.

El nuevo NetBeans IDE contiene las herramientas para que los desarrolladores de software puedan crear aplicaciones de Escritorio, enterprise, Web, y aplicaciones móviles, con el lenguaje Java, asi como también C/C++, PHP, JavaScript, Groovy, and Ruby.

3.4.4.2 QT CREATOR 1.3

Qt Creator es el sutil y multiplataforma Entorno Integrado de Desarrollo. Está diseñado fundamentalmente para hacer que el desarrollo en C++ sea más rápido y fácil. Se creó con el objetivo de usarlo para el desarrollo de múltiples plataformas como Windows XP y Vista, Linux y Mac OS X (38). Entre las principales características que este presenta es que tiene una biblioteca de clases intuitiva, herramientas de desarrollo integrado y presenta Integración con Visual Studio. Este IDE es portable e idóneo para aplicaciones futuras, y es principalmente por la utilización de la librería QT la cual ha sido utilizada en la realización de grandes proyectos. Esta biblioteca es multiplataforma para el desarrollo de aplicaciones gráficas. Qt es una aplicación integral y un marco para el desarrollo de aplicaciones de interfaz de usuario de Windows que también se puede implementar en muchos otros escritorios y sistemas integrados de explotación sin reescribir el código fuente. Dicho IDE presenta desventajas con respecto a NetBeans porque solo se emplea para el desarrollo de aplicaciones de escritorio, pero esto no es problema para el desarrollo del proyecto y aún así para el trabajo con medias está en ventaja por la utilización de QT.

A continuación se hace referencia a algunos elementos de esta biblioteca QT:

- **Klear**: Visualizador y grabador de TV digital.
- **JuK**: Reproductor de audio.
- **KAudioCreator**: Extractor digital de discos compactos.
- **KMPlayer**: Frontispicio para GStreamer, MPlayer o Xine.
- **Konverter**: Conversor de video, GUI para MEncoder.
- **KRadio**: Sintonizador de radio.
- **KsCD**: Reproductor de discos compactos.

3.4.3.4 IDEs PROPUESTOS

Después de un análisis sobre los IDE se llega a la conclusión de que se empleará QT Creator 1.3 con las librerías de QT, utilizando como lenguaje de programación C++ para las futuras aplicaciones de Escritorio del Departamento de Señales Digitales, ya que todos estos programas están basados en QT y están muy relacionados con las medias. Se propone además usar el IDE NetBeans para las aplicaciones Web y Escritorio por las grandes características que presenta, utilizando los lenguajes de programación PHP y Java. El IDE NetBeans respalda plenamente el desarrollo interactivo, por lo que los proyectos PHP siguen los patrones clásicos conocidos para los desarrolladores web. NetBeans es un conocido IDE para desarrollar en Java y también en PHP.

3.5 Sistema Gestor de Base de Datos (SGBD)

Es necesario proponer un Sistema Gestor de Base de Datos para crear y mantener una Base de Datos, asegurando su integridad y seguridad. Es imprescindible un gestor que le dé mantenimiento a la Base de Datos para que controle la manipulación de los datos y el acceso a estos.

3.5.1 POSTGRES SQL V-8.3

Este es un Sistema de Gestor de Bases de Datos Objeto-Relacionales (ORDBMS), basado en Software libre. Es ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo, donde cada usuario puede tener una visión sobre la última tarea que realizó. Ofrece nuevos conceptos como son: clases, herencia, tipos y funciones. Aporta en cierta forma potencia y flexibilidad

adicional como son las restricciones, los disparadores, las reglas y la integridad transaccional (35). Postgres SQL es altamente extensible, ya que soporta operadores y tipos de datos definidos por los distintos usuarios. Cuenta con un mejor soporte para los procedimientos que se acumulan en el servidor. Ofrece menor tiempo de respuesta a la hora de ejecutar cualquier consulta. Posee grandes características que tradicionalmente sólo se pueden ver en productos comerciales de alto calibre (41). Soporta transacciones y desde la versión 7.0, llaves foráneas (integridad referencial). Permite además que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

3.5.2 MYSQL

MySQL es un sistema gestor de bases de datos relacional (Relational Data Base Management System), multi–hilo y multi-usuario. Una base de datos relacional almacena datos en tablas separadas en lugar de poner todos los datos en un gran almacén, lo que añade velocidad y flexibilidad. Se trata de un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos. MYSQL trabaja en diferentes plataformas. Una sus principales ventajas es la velocidad que tiene a la hora de leer datos. Es muy popular en aplicaciones Web. Trabaja en numerosas plataformas como AIX, HP-UX, GNU/Linux, Mac OS X, Novell NetWare, OpenBSD, OS/2, Solaris, SunOS, y todas las versiones de Windows. Su mayor desempeño se logra cuando se combina con el lenguaje de programación PHP. Es gratuito para la mayor parte de los usos y su servicio de asistencia resulta económico. Mucho más rápido que la mayor parte de sus rivales. Se ejecuta en la inmensa mayoría de sistemas operativos y, la mayor parte de los casos, los datos se pueden transferir de un sistema a otro sin dificultad. Presenta una desventaja en cuanto a otros Sistemas Gestores de Base de Datos y es que es gratis para aplicaciones de código abierto, de lo contrario hay que pagar licencia comercial.

3.5.3 ORACLE

Oracle es también un sistema gestor de base de datos relacional. Es un sistema gestor robusto. Es considerado como uno de los sistemas de bases de datos más completos destacando soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma. Entre sus muchas características se puede resaltar que garantiza la

seguridad e integridad de los datos. Presenta estabilidad y gran escalabilidad. Las transacciones se ejecutan de forma correcta. Ayuda a administrar y almacenar grandes volúmenes de datos. La tecnología Oracle se encuentra prácticamente en todas las industrias alrededor del mundo. Garantiza además el funcionamiento de sus bases de datos. Aunque ha tenido mucha demanda en el mercado de servidores empresariales está sufriendo la rivalidad y competencia de Postgres SQL. Soporta además un subconjunto de SQL92 mayor que el que soporta MySQL.

3.5.4 Gestor de Base de Datos propuesto.

Se propone entonces Postgres SQL para la elaboración de un producto robusto y escalable. Se pudo observar que el gestor de BD Oracle es muy robusto y rápido, pero además es visible que PostgreSQL cuanto mayor es el número de registro y más compleja es la consulta, sus resultados son más rápidos. PostgreSQL es de código abierto. Es impresionante el gran soporte que tiene con la mayoría de los lenguajes de programación. Presenta una alta concurrencia lo que se encuentra entre sus principales características: mediante un sistema denominado MVCC (Acceso concurrente multiversión). Es estable, flexible. Se puede extender su funcionalidad además de tener gran compatibilidad con sistemas operativos.

3.6 Arquitecturas Propuestas.

Después de un estudio realizado sobre los proyectos del Departamento de Señales Digitales y de acuerdo con las características que estos presentan, se proponen el uso de diversas arquitecturas. Se seleccionó para el diseño de la arquitectura que se quiere proponer la Arquitectura Tres Capas, el patrón Modelo_Vista_Controlador y la Arquitectura Orientada a Objetos.

3.6.1 Arquitectura Tres Capas.

Si se quieren realizar aplicaciones de Escritorio en el Departamento de Señales Digitales, se propone hacer uso de la Arquitectura de Tres Capas. Tradicionalmente realizar aplicaciones compactas causa gran cantidad de problemas de integración en sistemas software que son complejos; ejemplos de estos sistemas pueden ser los de gestión de una empresa o los sistemas de información integrados en más de una aplicación. Dichas aplicaciones pueden presentar problemas de escalabilidad, seguridad etc. Para resolver este tipo de problema se asumió la división de las

aplicaciones en capas, las cuales se mencionan a continuación. Si se establece una separación entre capas se obtiene una potente arquitectura.

Capas: ayudan a estructurar aplicaciones que pueden estar dispersos en grupos de subtareas, en las cuales cada grupo de subtarea está en un cierto nivel de abstracción. Cuando se utiliza el término capa se hace referencia a cómo una solución se encontrará dividida desde un punto de vista lógico.

Modelo de Tres Capas

La principal función de este modelo es crear una modularidad del sistema de forma tal que asigne a cada capa una funcionalidad específica y bien definida; además de localizar errores y mejorar considerablemente el soporte del mismo. Las interacciones entre las capas ocurren generalmente por invocación de métodos(Garlan y Shaw). Por definición, los niveles de abajo no deben poder utilizar funcionalidad ofrecida por los de niveles superiores.

En este patrón de arquitectura cada capa le proporcionará servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior. Al dividir un sistema en capas, cada una de ellas puede tratarse de forma independiente, sin tener que conocer los detalles de las demás.

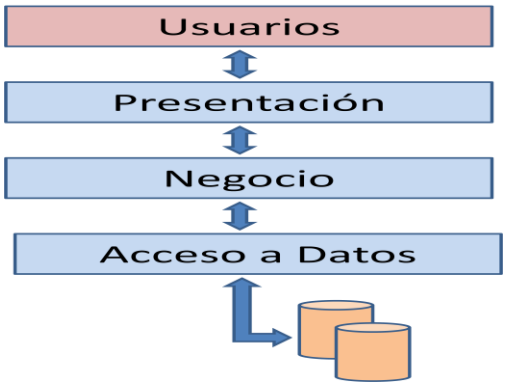

Ilustración 1 Modelo de Tres Capas

La capa de Presentación

La arquitectura en 3 capas está compuesta por una capa de presentación o también denominada capa superior. Es la encargada de controlar las formas de navegación de los usuarios por las pantallas. Reúne todos los aspectos del software que tiene que ver con las interfaces y la interacción con los diferentes tipos de usuarios. Estos incluyen

el manejo y aspecto de las ventanas, el formato de los reportes, menús, gráficos y elementos multimedia en general. La capa de presentación es una especie de intermediaria entre el usuario y el sistema, es como una ventana que captura la información de un usuario realizando un filtrado previo para comprobar que no existen errores de formato y le comunica la información que este solicita. Esta capa solamente se comunica con la capa continua a ella, la capa de negocio.

La capa de Negocio

Conforma todas las características del software que apoyan los procesos de negocio que llevan a cabo los usuarios. Es en ella donde se encuentran los programas que se ejecutan, estableciéndose además las reglas, las validaciones y los cálculos que un sistema debe efectuar. Es responsable de la implementación de las reglas del negocio. Se conoce como capa intermedia; la cual recibe la petición del usuario a través de la capa de presentación y se encarga de responderle mediante repositorios de información. Esto se traduce a la comunicación que tiene con la capa de presentación para recibir solicitudes y mostrar resultados. Se comunica además con la capa de datos, en este caso para solicitar al gestor de base de datos acciones tales como almacenar o recuperar datos de él.

La capa de Acceso a Datos

Reune de cierta forma todas las características del software que tienen relación con el manejo de los datos persistentes, por lo que también se le llama capa de Bases de Datos, existiendo especificaciones donde se maneja cómo será la comunicación entre cada una de las capas definidas. Es la representación real de los datos. En ella se representan los procedimientos almacenados y el esquema de los datos. Está compuesta por uno o varios gestores de Bases de Datos que realizan el almacenamiento de toda la información, recibiendo solicitudes de almacenamiento o recuperación desde la capa de negocio.

3.6.1.1 ¿ POR QUÉ SE PROPONE LA ARQUITECTURA EN CAPAS ?

La división de un sistema en capas facilita el diseño arquitectónico, permitiendo la construcción de sistemas débilmente acoplados, lo que significa que si se reducen las dependencias entre las capas resultaría más fácil la implementación de cada capa por separado sin afectar al resto del sistema.

Entre las principales ventajas que hacen que se proponga esta arquitectura para aplicaciones de Escritorio se encuentran:

- Modularidad del sistema.
- Mejora el soporte del sistema.
- 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.
- Facilita la localización de errores.
- Permite el desarrollo paralelo del sistema por cada capa.
- Mayor flexibilidad (se pueden añadir módulos para dotar al sistema de otras funcionalidad).
- Facilita el mantenimiento del proyecto.
- Proporciona amplia reutilización. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.

Es válido mencionar que para el desarrollo de una aplicación de Escritorio es importante este patrón pues el mismo permite a los programadores corregir sus errores con más facilidad debido a la estructura que tendrá el código.

Para la capa de *interfaz de usuario* es recomendable usar el framework Swing. Este framework está compuesto por un amplio conjunto de componentes. En el caso de la capa de *lógica del negocio*, se propone el framework Spring, el cual prporcionará una potente configuración de la aplicación que esté basada en el lenguaje de programación Java con el IDE NetBeans. Spring facilita el desarrollo de otras funcionalidades. Permite que los objetos de acceso a datos y del negocio sean reutilizables. Se basa fundamentalmente en permitir que los objetos del negocio y de acceso a datos sean reusables. Facilita la manipulación de los objetos y mejora la práctica de la programación. Para la capa de *acceso a datos* se propone utilizar el framework Hibernate en el caso de utilizar el lenguaje de programación JAVA.

3.6.2 PATRÓN DE ARQUITECTURA MODELO_VISTA_CONTROLADOR (MVC).

La *aplicaciones Web* en el Departamento de Señales Digitales se quieren modelar haciendo uso del patrón Modelo_Vista _Controlador (MVC) con el objetivo de utilizar la separación de las responsabilidades de cada una de las capas que lo componen y así lograr mayores facilidades de desarrollo. Este patrón se utiliza principalmente cuando es necesario modularizar la interfaz de usuario y las reglas del negocio.Se ve frecuentemente en aplicaciones de este tipo.

Dicho patrón es muy utilizado en el diseño de aplicaciones que contengan sofisticadas interfaces como lo son las aplicaciones Web. Es usado con frecuencia en aplicaciones que necesiten la habilidad de mantener múltiples vistas de la información. Normalmente, la vista cambia más veces que el modelo y el controlador, por tanto, si se realiza un diseño que mezcle los componentes de la vista y controlador entonces esto traerá consigo que cada vez que se necesite cambiar la vista, habrá que modificar trabajosamente los componentes del controlador, lo que conlleva a mayores posibilidades de cometer errores y a un incremento de trabajo para los desarrolladores. Es por eso que se necesita hacer uso de ese patrón de arquitectura en este tipo de aplicaciones, para desacoplar la vista del modelo, con la finalidad de mejorar la reusabilidad, de forma tal que las modificaciones en la vista impacten en menor medida en el modelo o el controlador.

El patrón de arquitectura MVC se caracteriza por dividir una aplicación interactiva en tres componentes. Este modelo contiene los datos y funcionalidades de fondo. Se utiliza principalmente cuando es necesario modularizar la interfaz de usuario, las reglas de negocio y el control de eventos.

Se caracteriza por separar el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres partes diferentes (32):

- **Modelo**
- **Vista**
- **Controlador**

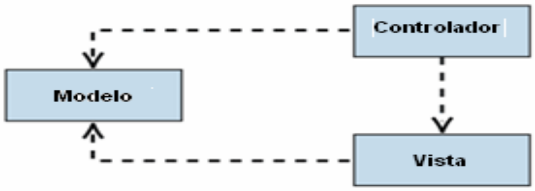

Ilustración 2 Estilo Modelo_Vista_Controlador

Modelo

Administra el comportamiento y los datos del dominio de aplicación, responde a requisitos de información sobre su estado(usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado(habitualmente desde el controlador). Mantiene el conocimiento del sistema. Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada. No depende de ninguna vista y de ningún controlador.

El modelo es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar datos.

Vista

Maneja la visualización de la información. Presenta el modelo en un formato adecuado para interactuar con el usuario, dicha capa es conocida como: interfaz de usuario.

Controlador

Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado. Tiene tres variantes principales: Activa, Pasiva y Documento-Vista. Esta parte responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Recibe además las entradas, traducidas a solicitudes de servicio para el modelo.

3.6.2.1 ¿ POR QUÉ SE PROPONE MODELO_VISTA_CONTROLADOR (MVC) ?

El patrón de arquitectura Modelo_Vista _Controlador se utiliza principalmente cuando es necesario modularizar la interfaz de usuario, las reglas del negocio y el control del negocio. Entre las principales ventajas que este presenta y que hacen que se proponga esta arquitectura se encuentran:

- Evita poner el código indebido en la capa impropia.
- Facilita despliegue en caso de modificaciones en el modelo de datos.
- Se puede mostrar distintas variantes de interfaz gráfica simultáneamente.
- La interfaz tiende a cambiar más rápido que las reglas del negocio. Agregar otros tipos de vista no afecta el modelo.

El patrón MVC es de gran utilidad para aplicaciones Web, ya que permite una mejor organización del código haciendo que el sistema sea más modular. Si se utiliza este patrón se posibilita la disminución de la complejidad del sistema, lo que sería más fácil para los programadores.

Para las *aplicaciones Web* se propone además para la abstracción de datos *DOCTRINE:*

PROPEL es el ORM clásico de Symfony. Su principal ventaja es que está completamente integrado con Symfony y que decenas de plugins sólo funcionan para Propel. Su desventaja es que la versión 1.2 que se utiliza actualmente tiene menor rendimiento que las versiones 1.3 y 1.4 donde se utiliza DOCTRINE por defecto.
DOCTRINE es el ORM del futuro de Symfony. Su principal ventaja es el rendimiento en ejecución y la forma tan concisa en la que se pueden escribir consultas muy complejas.

3.6.3 ARQUITECTURA ORIENTADA A OBJETOS.

En los sistemas de información diseñados bajo el paradigma orientado a objetos es recomendable el uso de esta arquitectura. La arquitectura orientada objetos será utilizada fundamentalmente donde se emplee una programación orientada a objetos.

Esta arquitectura se encuentra presente en aplicaciones *Web* y de *Escritorio*, es por tanto que surge como otra propuesta para los productos del Departamento de Señales Digitales.

Los componentes de este tipo de esta arquitectura son los objetos. Estos objetos representan una especie de componentes que se les llama managers, debido a que

son responsables de preservar la integridad de su propia representación (Garlan y Shaw).

Los componentes de este estilo se basan en principios orientados a objetos, los cuales se refieren a encapsulamiento, herencia y polimorfismo (32). Dichos objetos no pueden ser accesibles desde otros objetos. Los objetos y sus interacciones son el centro en el diseño de la arquitectura orientada a objetos y en la estructura de la aplicación. Un objeto será ante todo una entidad reutilizable en el entorno de desarrollo de cualquier sistema.

Una de las características que presenta esta arquitectura es que las clases interfaces están independientes o separadas de sus respectivas implementaciones, lo que lleva a la conclusión de que la distribución de estos objetos será totalmente transparente.

Los objetos se relacionan a través de la llamada de métodos y mensajes. Existen diferentes variantes de este estilo, ya que algunos sistemas, pueden admitir que los objetos sean tareas concurrentes; mientras que otros permiten que los objetos posean múltiples interfaces facilitando un distinto comportamiento.

3.6.3.1 ¿ POR QUÉ SE PROPONE LA ARQUITECTURA ORIENTADA A OBJETOS ?

Entre las principales ventajas que influyen en la propuesta de esta arquitectura se encuentran:

- El refinamiento o descomposición funcional es sencillo.
- El encapsulamiento y reutilización, es de fácil descomposición en una colección de agentes interactivos.
- Se puede modificar la implementación de un objeto sin afectar a sus clientes.

Esta arquitectura aunque para muchos sistemas es invisible, está en casi todas las aplicaciones, puesto que están basados en una programación orientada a objetos y en la cual se emplea fundamentalmente la *herencia y el polimorfismo*.

3.7 Protocolos de comunicación.

Un protocolo es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red. Un protocolo es una convención o estándar que controla o permite la conexión, comunicación, y transferencia de datos entre dos puntos finales. En su forma más simple, un protocolo puede ser definido como las reglas que dominan la sintaxis, semántica y sincronización de la comunicación.

3.7.1 PROTOCOLOS DE COMUNICACIÓN ENTRE APLICACIONES WEB.

3.7.1.1 HTTP

Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto). HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (un navegador web) se lo conoce como "user agent" (agente del usuario). A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL). Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

3.7.1.2 HTTPS

Hypertext Transfer Protocol Secure (en español: Protocolo seguro de transferencia de hipertexto), más conocido por sus siglas HTTPS. Es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP. La idea principal de https es la de crear un canal seguro sobre una red insegura.

El protocolo HTTPS utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP. Cabe mencionar que el uso del protocolo HTTPS no impide que se pueda utilizar HTTP. Es aquí, cuando nuestro navegador advertirá sobre la carga de elementos no seguros (HTTP), estando conectados a un entorno seguro (HTTPS).

Secure Sockets Layer (SSL): Seguridad de la Capa de Transporte. Es un protocolo criptográfico que proporciona comunicaciones seguras en Internet. SSL proporciona autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de la criptografía. Habitualmente, sólo el servidor es autenticado (es decir, se garantiza su identidad) mientras que el cliente se mantiene sin autenticar; la autenticación mutua requiere un despliegue de infraestructura de claves públicas (o PKI) para los clientes. Los protocolos permiten a las aplicaciones cliente-servidor comunicarse de una forma diseñada para prevenir escuchas (eavesdropping), la falsificación de la identidad del remitente (phishing) y mantener la integridad del mensaje. SSL implica una serie de fases básicas:

- Negociar entre las partes el algoritmo que se usará en la comunicación.
- . Intercambio de claves públicas y autenticación basada en certificados digitales.
- Cifrado del tráfico basado en cifrado simétrico.

3.7.1.2 FTP

File Transfer Protocol o FTP (en español Protocolo de Transferencia de Archivos). Es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor. Permite que un equipo cliente se pueda conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

El Servicio FTP es ofrecido por la capa de Aplicación del modelo de capas de red TCP/IP al usuario, utilizando normalmente el puerto de red 20 y el 21. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el login y password del usuario en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún tipo de cifrado, con lo que un posible atacante puede capturar este tráfico, acceder al servidor, o apropiarse de los archivos transferidos.

3.7.1.3 SOAP

SOAP (Simple Object Access Protocol) es un protocolo estándar que permite la comunicación y la interoperabilidad entre diversas aplicaciones Web desarrolladas bajo tecnologías diferentes. El objetivo primordial de SOAP es Establecer un protocolo estándar de invocación de servicios remotos, basado en protocolos estándares de Internet: HTTP (Protocolo de transporte de Hipertexto) para la transmisión y XML (lenguaje de marcado extensible) para la codificación de datos.

3.7.2 PROTOCOLOS DE COMUNICACIÓN ENTRE APLICACIONES DE ESCRITORIO.

3.7.2.1 TCP/IP

El conjunto TCP/IP está diseñado para enrutar y tiene un grado muy elevado de fiabilidad, es adecuado para redes grandes y medianas, así como en redes empresariales. Se utiliza a nivel mundial para conectarse a Internet y a los servidores web. Es compatible con las herramientas estándar para analizar el funcionamiento de la red. Es rápido en redes con un volumen de tráfico grande donde haya que enrutar un gran número de tramas.

Este protocolo se encargará de que la comunicación sea posible. TCP/IP es compatible con cualquier sistema operativo y con cualquier tipo de hardware. Cualquier máquina de la red puede comunicarse con otra distinta y esta conectividad permite enlazar redes físicamente independientes en una red virtual llamada Internet. Las máquinas en Internet son denominadas "hosts" o nodos. TCP/IP. Proporciona la base para muchos servicios útiles, incluyendo correo electrónico, transferencia de ficheros y login remoto. El correo electrónico está diseñado para transmitir ficheros de texto pequeños. También pueden proporcionar chequeos de seguridad controlando las transferencias. El login remoto permite a los usuarios de un ordenador acceder a una máquina remota y llevar a cabo una sesión interactiva.

3.7.3 PROTOCOLOS DE COMUNICACIÓN ENTRE UNA APLICACIÓN Y SU SISTEMA GESTOR DE BASE DE DATOS.

3.7.3.1 ODBC(OPEN DATABASE CONNECTIVITY)

Es un estándar de acceso a bases de datos que utilizan los sistemas Microsoft. A través de ODBC, en un sistema Windows se puede conectar con cualquier base de datos. Permite conectar con cualquier base de datos de la que exista un driver ODBC.

Los creadores de las distintas bases de datos son los responsables de crear un driver ODBC para que su base de datos se pueda conectar desde un sistema Microsoft.

El objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos almacene los datos, ODBC logra esto al insertar una capa intermedia llamada manejador de Bases de Datos, entre la aplicación y el DBMS, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el DBMS entienda.

Para conectar con ODBC una base de datos se ha de crear un DSN, que es un nombre que se asocia a una conexión por ODBC para referirse a ella desde las aplicaciones o programas que deban conectarse con la base de datos.

3.7.3.2 JDBC (JAVA DATABASE CONNECTIVITY)

JDBC es el acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos ([URL]) que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión, para ello provee en localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar con cualquier tipo de tareas con la base de datos a las que tenga permiso: consultas, actualizaciones, creación, modificación y eliminación de tablas, ejecución de procedimientos almacenados en la base de datos, etc. Es importante resaltar que la conectividad y apertura de controles permitiría una mejor interconexión de sistemas heterogéneos. En la actualidad existen ODBC para muchos sistemas de bases datos, tales como Informix, Access, PostgreSQL, MySQL, Oracle y SQL Server.

3.8 Tecnologías telemáticas y de servicios.

Dichas tecnologías se emplearán como parte del desarrollo de software en el Departamento de Señales Digitales. Además facilitan el proceso de desarrollo y perfeccionamiento de cualquier aplicación que se desee implementar.

3.8.1 TECNOLOGÍAS TELEMÁTICAS Y DE SERVICIOS PROPUESTAS.

- Controlador de versiones “Subversion (SVN)”.
- ALFRESCO como aplicación de Gestión Documental.
- Algún Portal que necesite determinado producto que se esté desarrollando.
- Aplicaciones para el control de tareas en el proyecto.

3.8.1.1 CONTROLADOR DE VERSIÓN SUBVERSION (SVN).

Es un sistema de control de versiones libre y de código abierto. El mismo usa como componente fundamental los llamados repositorios a los cuales se puede acceder para adquirir o guardar información. Estos repositorios son servidores de ficheros que guardan todos los cambios que se le hagan a estos ficheros. Permiten además recuperar versiones antiguas de los datos o explorar el historial de cambios de los mismos. Se puede acceder a ellos a través de la red, lo cual se usará por personas que estén en diferentes lugares físicamente, así como administrar y realizar cambios a los datos.

3.8.1.2 ALFRESCO COMO APLICACIÓN DE GESTIÓN DOCUMENTAL.

Este tipo de aplicaciones permite la gestión de documentos de apoyo a los productos que se desarrollan en la Universidad de las Ciencias Informáticas. Es de gran importancia, ya que lleva a cabo la documentación organizada de todo lo referente al/los producto(s) que se están desarrollando.

A continuación se mencionan algunos ejemplos de cómo aprovechar esta tecnología de servicio en beneficio del desarrollo de los proyectos que se llevan a cabo:

- Permite controlar las evaluaciones de los desarrolladores de un producto.

- Se puede llevar a cabo el parte de asistencia de cada uno de los integrantes del proyecto.

- Se puede llevar a cabo el control del horario que cumple cada desarrollador.

- Permite llevar a cabo lo relacionado con el cronograma de trabajo de los desarrolladores de un producto software.

3.8.1.3 PORTAL.

Proporcionar un *Portal* para los productos del Departamento de Señales Digitales sería enriquecedor tanto para los desarrolladores como para las personas ajenas a este. A continuación se mencionan algunos ejemplos de elementos de los que podría disponer este *Portal* que se confeccione:

- Publicar noticias de cada producto del Departamento.

- Mostrar algún tipo de ranking entre los productos.

- Mostrar algún tipo de rancking entre desarrolladores de cada producto.

- Informar sobre los avances de cada producto para motivar al trabajo de cada desarrollador.

- Dar a conocer la historia de cada producto, por qué es necesario realizarlo, por qué es importante para la Universidad y para el País.

3.8.1.4 APLICACIÓN PARA EL CONTROL DE TAREAS.

Se considera una buena opción que cada producto o proyecto tenga un pequeño sitio donde se publiquen las tareas a las que se les debe dar cumplimiento. La universidad de manera particular propone la aplicación REDMINE para mostrar este tipo de elementos. A continuación se muestran algunos ejemplos de elementos que podría contener este sitio que se realice:

- Publicar las tareas que debe cumplir cada desarrollador del proyecto.

- Mostrar quién va más atrazado o adelantado en su trabajo.
- Mostrar el tiempo en que cada desarrollador le dio cumplimiento a su tarea.
- Publicar el grado de cumplimiento de cada tarea.

3.3.9 CONCLUSIONES PARCIALES.

En este capítulo se abordaron diversos temas como es lo referente a la selección de los diversos tipos de arquitecturas existentes que se proponen para aplicaciones Web y Escritorio, así como tecnologías telemáticas y de servicios y herramientas necesarias para el desarrollo de un proyecto, todo esto con el objetivo de desarrollar mejores aplicaciones en el Departamento de Señales Digitales. Se definieron las herramientas de modelado y de desarrollo que se usarán en el desarrollo de los productos del departamento. Se tuvo en cuenta para su selección que dichas tecnologías debían ser fáciles de usar y compatibles con varias plataformas, etc.

Conclusiones generales

En el presente trabajo se realizó un estudio detallado de los principales aspectos de la descripción arquitectónica. Se realizó una propuesta de atributos calidad idóneos para medir la complejidad de la calidad de un software.

Se evaluaron las metodologías, tecnologías y tendencias actuales, lo que permitió la selección de las herramientas que más se adecuan a las necesidades para el desarrollo de aplicaciones en el departamento de Señales Digitales.

A través de la descripción de las arquitecturas se concluyó que para aplicaciones Web se utilizarán el patrón Modelo_Vista_Controlador para separar la vista del modelo y el controlador, con el objetivo de lograr mayor modularidad en el sistema. Se propuso una arquitectura Tres Capas para independizar cada una de las capas y así facilitar el desarrollo e implementación de las aplicaciones de Escritorio.

Se definieron protocolos de comunicación para establecer el intercambio de datos entre las aplicaciones de Escritorio, Web y sus Bases de Datos.

Se establecieron tecnologías telemáticas y de servicios para el proceso de desarrollo de un software con el objetivo de que sirvan de apoyo y motivación a los desarrolladores de estos.

Se cumplieron los objetivos trazados en esta investigación y a la vez se dio una nueva propuesta de una arquitectura para los productos del Departamento de Señales Digitales.

RECOMENDACIONES

Se recomienda aplicar las arquitecturas propuestas en las futuras aplicaciones que se construyan en el Departamento de Señales Digitales para obtener productos competentes en el mercado internacional.

BIBLIOGRAFÍA

1. *Arquitectura del Software*.Parte 1. Introducción

2. **Armando Arce O**. *Sistemas Distribuidos*.Febrero 2000.

3. **Carlos E. Cuesta.** *Arquitecturas del Software*. Universidad Rey Juan Carlos. Ingeniería del software 1

4. **Garlan, David y Shaw Mary**. "*An introduction to software architecture*". CMU Software Engineering Institute Technical Report, CMU/SEI-94-TR-21, ESC-TR-94-21, 1994.

5. **Paul Clements**. *A Survey of Architecture Description Languages*. Alemania 1996

6. **Sommerville, Ian**. *Ingenieria del software. Un enfoque práctico.*Madrid 2005.

7. **Tanenbaum, A. S.** *Sistemas Operativos distribuidos*. Ed Prentice Hall. 1ª Ed. (1996).

8. *Departamento de las Ciencias de Comunicación y Software*, Carnegie Mellon University, Pittsburgh, Pensilvania, EE-UU ,1989.

9. http://empresas.telefonica.es/documentacion/WP_Web_Services.pdf

10. http://html.rincondelvago.com/protocolos-de-comunicacion_1.html

11. http://es.kioskea.net/contents/internet/tcp.php3

12. http://www.augcyl.org/?q=glol-intro-sistemas-distribuidos [Colouris 1994]

13. http://html.rincondelvago.com/sistemas-distribuidos.html (12-12-09)

14. **Greimas y Courtés.** *"Texto" en Diccionario razonado de la teoría del lenguaje.* Madrid: Editorial Gredos. ISBN 84-249-08551-1. (409,410).

*15***. Iglesias Simón; Pablo***; "El diseñador de sonido: función y esquema de trabajo",* ADE-Teatro Nº 101. Julio-agosto de 2005. Páginas 199-215.

16. **McMillan, S.J.** (2002). *Exploring Models of Interactivity from Multiple Research Traditions:* Users, Documents, And Systems. In L. Lievrouw and S. Livingston (Eds.), Handbook of New Media (pp. 162-182). London: Sage.

17. **Rafaeli, S.** (1988). *Interactivity:* From new media to communication. In R. P. Hawkins, J. M. Wiemann, & S. Pingree (Eds.),Sage Annual Review of

Communication Research: Advancing Communication Science: Merging Mass and Interpersonal Processes, 16, 110-134. Beverly Hills: Sage.

18. **Bou Bauzá Guillem** (2003).*El guión Multimedia.* Anaya Multimedia (Eds.), Madrid.

19. **VAUGHAN***, Tay. Todo el poder de la Multimedia.* Segunda Edición. Editorial Mc Graw Hill. México. 1994.

20. http://www.infovision.com.mx

21. **M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts.** *Refactoring: Improving the Desing of Exist Code.* s.l. : Addison-Wesley , 1999.

22. **Reynoso, Carlos y Kicillof, Nicolás.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft* . Universidad de Buenos Aires : s.n.

*23.***Gamma, Erich,** y otros*. Desing Patterns.* Elements of Reusable Object-Oriented Software(GoF-Group of Four). 0201633612.

*24.***SEMATECH, Austin.** *Computer Integred Manufacturing(CIM)* Framework Specification- Version 2.0. 1998 : s.n. 93061697j.

25. **Fayad, Mohamed, Scmidt, Douglas y Johndson, Ralph***. Building Application Frameworks:* Objects- Oriented Foundations of Framework Desing . 1999.

*26.***Christopher, Alexander.** *A Pattern Language.* Oxford, Universuty Press : s.n., 1977.

27. **Anay Carrillo Ramos,** *http://www.eumed.net*

28. **Morales, Mauricio Venegas***. Captura y transmisión de video en GNU/Linux.*

29. **Jackson Colares de Silva,** *Elsonido en la multimedia: la importancia de la producción del audio en los diseños de materiales multimedia para la enseñanza,http://www.filos.unam.mx.*

30. **Erika Camacho, fabio Cardeso, Gabriel Núñez***, Arquitecturas de software: Guia de estudio,* abril 2004.

31. **Cervera Paz, Angel**, *El modelo de McCall como aplicación de la calidad a la revisión del software de gestión empresarial*, Dpto. Lenguajes y Sistemas Informáticos, Universidad Cádiz.

32. **Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M** (1996). *Pattern-Oriented Software Architecture: A System of Patterns. John Wiley*

33. **Gamma, Erich, y otros.** *Desing Patterns. Elements of Reusable Object-Oriented Software(GoF-Group of Four).*

34. **SEMATECH, Austin**. *Computer Integred Manufacturing(CIM) Framework Specification- Version 2.0* (1998)

35. **Lockhart, Thomas***. The PostgreSQL Development Team. s.l. : Thomas Lockhart, (1996).*

36. **. Rosenblum, Nenad Medvidovic y David S**. *Domains of Concern in Software Architectures and Architecture Description Languages,*California(1997).

37**. CRAIG WALLS, Ryan Breidenbach.** *Spring in Action. Second Edition. United States of America*. (2008).

38. **Gallardo,David, Burnette, Ed and McGovern , Robert.** *Eclipse in Action: A Guide for Web Developers.(*2003).

39. **S.A, Grupo Soluciones Innova***. GSI. [Citado el: 12 de mayo de 2010.] http://www.rational.com.ar/apertura/acerca.html.*

40. **Fabien Potencier, François Zaninotto.** *Symphony, La Guia Definitiva.(2008)*

41. PostgreSQL Comunitario Mexico¿Que es PostgreSql?
Disponible en: http://www.postgresql.org.mx/?q=node/6. (2010)

42. *http://www.milestone.com.mxb [Citado el: 10 de mayo de 2010.]*

43. http://www.slideshare.net [Citado el: 10 de mayo de 2010.]

44. http://e-articles.info [Citado el: 1 de mayo de 2010.]

*45***. Sergio Luján Mora***.C++ paso a paso. http://gplsi.dlsi.ua.es [Citado el: 20 de mayo de 2010.]*

46.Symfony en pocas palabras. http://librosweb.es [Citado el: 22 de mayo de 2010].

47. http://www.juntadeandalucia.es [Citado el: 10 de mayo de 2010].

*48***. Ing. José Joskowicz**. *Reglas y Prácticas en eXtreme Programming.* (2008)

GLOSARIO DE TÉRMINOS

PTARTV: Plataforma de transmisión abierta para radio y televisión

IEEE: Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos).

Lenguaje de Programación: Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis.

AS: Arquitectura de Software

Capas: Es uno de los patrones de diseño más utilizado para cualquier tipo aplicaciones, en este, básicamente se divide los elementos de diseño en paquetes.

Herramientas CASE (Computer Aided Software Engineering): Se incluyen una serie de herramientas, lenguajes y técnicas de programación que permiten la generación de aplicaciones de manera semiautomática.

RUP: Metodología de desarrollo de software basada en UML. Organiza el desarrollo de software en 4 fases.

Framework: Marco de trabajo, solución reutilizable y extensible.

ORM: Mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

APIs: es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Usados generalmente en las bibliotecas.

WSDL: son las siglas de *Web Services Description Language*, un formato XML que se utiliza para describir servicios Web