ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

INTEGRACIÓN DE LA METODOLOGÍA ÁGIL XP EL ESTUDIO DE WINDOWS COMMUNICATION FOUNDATION COMO ALTERNATIVA METODOLÓGICA PARA EL DESARROLLO DE SOFTWARE ORIENTADO A SERVICIOS CASO PRACTICO CIPFIE-ESPOCH

TESIS DE GRADO

Previa obtención del título de

INGENIERO EN SISTEMAS INFORMÁTICOS

Presentado por:

LUIS ALBERTO RIVERA RIERA

JUAN PABLO TIXE ALULEMA

RIOBAMBA – ECUADOR

2009

AGRADECIMIENTO

Al culminar esta tesis de grado queremos expresar nuestro profundo agradecimiento al Ing. Danilo Pastor como director, a la Ing. Gloria Arcos e Ing. Ivan Menes como asesores; quienes con su apoyo ayudaron a la exitosa culminación de la presente.

A Maria Augusta Larrea, por todas las facilidades prestadas en el CIPFIE y a las demás personas que de una u otra manera colaboraron durante el desarrollo y culminación de este trabajo.

DEDICATORIA

Dedico la culminación de esta Tesis de Grado a
mi padre Luis Alberto Rivera Pozo,
a mi madre Blanquita Riera
y a mis hermanas Paulina, Daniela y Gina
quienes me supieron dar su apoyo, amor
y cariño durante todo este tiempo y en una
forma muy especial a DIOS por la vida que me
ha regalado.

Luis Alberto Rivera Riera

A Dios por la oportunidad de llegar a la meta.
A Enma y Livia por todo su amor y apoyo
incondicional.
A Mayri por brindarme ese empuje y equilibrio
necesario en mi vida.
A Pablito mi inspiración diaria.
A Vicente y Livia, hermanos casi hijos.
A Doña Loly, Don Pepe, Didi, Angy que siempre
están pendientes de nuestras acciones.

Juan Pablo Tixe Alulema

NOMBRE	FIRMA	FECHA
Dr. Ms.C. Romeo Rodríguez **DECANO FACULTAD DE INFORMATICA Y ELECTRONICA**	……	……
Ing. Iván Menes **DIRECTOR DE LA ESCUELA DE INGENIERIA EN SISTEMAS**	……	……
Ing. Danilo Pastor **DIRECTOR DE TESIS**	……	……
Ing. Gloria Arcos **MIEMBRO DEL TRIBUNAL**	……	……
Ing. Iván Menes **MIEMBRO DEL TRIBUNAL**	……	……
Lcdo. Carlos Rodríguez **DIRECTOR CENTRO DE DOCUMENTACIÓN**	……	……

NOTA DE LA TESIS ……

"Nosotros, Luís Alberto Rivera Riera y Juan Pablo Tixe Alulema somos responsables de las ideas, doctrinas y resultados expuestos en esta tesis; y el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO."

Luís Alberto Rivera Riera.

Juan Pablo Tixe Alulema.

ÍNDICE DE ABREVIATURAS

API. Application Programming Interface (Interfáz de programas de aplicación).

ASP. Active Server Pages - Páginas de servidor activas.

COM. Modelo de Objetos y Componentes.

CLR. Common Language Runtime.

CIPFIE. Centro de Investigación y Producción de la Facultad de Informática y Electrónica.

COM+. .NET Enterprise Services.

DCOM. Distributed COM (COM Distribuido).

DLLs. Dynamic-Link Library.

DTC. Controlador de transacciones distribuidas.

FIE. Facultad de Informática y Electrónica.

GAC. Global Assembly Cache.

HTTP. Hipertext Transport Protocol

HTML. Hipertext Markup Languaje (Lenguaje de marcado de hipertexto).

IDL. Interface Description Language.

IIS. Internet Information Services.

IPSec. Internet Protocol Security.

IT. Tecnología informática.

IU. Interface User.

IDE. Entorno Integrado de Desarrollo

JIT. Just-In-Time.

MSIL. Lenguaje Intermedio de Microsoft.

MSMQ. Microsoft Message Queuing

Q.S. Calidad de Servicio

RPC. Remote procedure calls.

SDK. Software Development kit.

SOA. Arquitectura Orientada a Servicios.

SOAP. Simple object access protocol.

SQL. Structured Query Language - Lenguaje estructurado de consulta.

SSL. Secure Sockets Layer (Capa de sockets seguros).

TCP / IP. Transfer Control Protocol / Internet Protocol.

UDDI. Universal description discovery and integration.

URI. Uniform Resource Identifier (Identificador uniforme de recursos).

WAS. Windows Activation Service.

WSDL. Lenguaje de descripción de servicios web.

W3C. World Wide Web Consortium.

WS-I. Web Services Interoperability Organization.

WSE. Web Services Extensions.

XP. Extreme Programming

WCF. Windows Communication Foundation.

ÍNDICE GENERAL

CAPITULO II

2. ESTUDIO DEL PARADIGMA ARQUITECTÓNICO DE ORIENTACIÓN A SERVICIOS Y DE WINDOWS COMMUNICATION FOUNDATION (WCF)

CAPITULO III

3. ESTUDIO DE LA METODOLOGÍA ÁGIL XP

CAPITULO IV

4. PROPUESTA METODOLÓGICA PARA EL DESARROLLO DE SOFTWARE ORIENTADO A SERVICIOS INTEGRANDO A LA METODOLOGÍA ÁGIL XP EL ESTUDIO DE WINDOWS COMMUNICATION FOUNDATION

CAPITULO V

5. CASO PRACTICO: DESARROLLO DEL SISTEMA INFORMÁTICO DE GESTIÓN DE PROYECTOS DE GRADO DEL CIPFIE-ESPOCH

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El negocio cada vez exige crear aplicaciones más complejas, con menor tiempo y presupuesto. Crear estas aplicaciones, requiere en muchos casos de funcionalidades ya antes implementadas como parte de otros sistemas.

Una estrategia de aplicaciones empresariales debe facilitar su integración. En la actualidad la orientación a servicios es una nueva forma de modelar el desarrollo de software, permitiendo considerar a los servicios como unidades de creación de software, logrando así una fácil integración y desarrollo.

El modelado orientado a servicios requiere de actividades adicionales que no son encontrados en un tradicional modelado orientado a objetos, un enfoque más adecuado consiste en modelar las aplicaciones orientadas a servicios específicamente con metodologías orientadas a servicios. Surgiendo la necesidad de una metodología que guíe el desarrollo de un software orientado a servicios.

El presente trabajo de investigación pretende proponer una guía metodológica para el desarrollo de aplicaciones orientadas a servicios tomando como base el estudio de la metodología ágil XP y el modelo de programación unificada de Microsoft denominado Windows Communication Foundation (WCF).

Para lograr nuestro objetivo, en el Capitulo II vamos a realizar el estudio del Paradigma "Arquitectónico de la Orientación a Servicios" y el modelo de programación unificada de "Windows Communication Foundation", permitiendo entender el modelado, los principios y como implementar la orientación a servicios con "Windows Communication Foundation".

En el Capitulo III nos concentraremos en el estudio de la metodología ágil XP la cual nos servirá como base para nuestra propuesta metodológica.

En el capitulo IV expondremos nuestra propuesta metodológica resultado del estudio de los capítulos anteriores. Pero a más de exponer nuestra propuesta metodológica, en el capitulo V desarrollaremos el sistema SIGEP (Sistema de gestión de proyectos) basado en la metodología propuesta, el cual permitirá gestionar los proyectos de tesis de la Facultad de Informática y Electrónica administrados por el CIPFIE.

CAPITULO I

MARCO DE REFERENCIA

INTRODUCCIÓN

SOA es un modelo de componentes que interrelaciona las diferentes unidades funcionales de las aplicaciones, denominadas servicios, a través de interfaces y contratos bien definidos entre esos servicios. La interfaz se define de forma neutral, independiente de la plataforma hardware, del sistema operativo y del lenguaje de programación utilizado. Esto permite a los servicios, construidos sobre sistemas heterogéneos, interactuar entre ellos de una manera uniforme y universa.

En la actualidad la orientación a servicios es una nueva forma de modelar el desarrollo de software, permitiendo considerar a los servicios como unidades de creación de software, logrando así una fácil integración y desarrollo.

El modelado orientado a servicios requiere de actividades adicionales que no son encontrados en un tradicional modelado orientado a objetos, un enfoque más adecuado consiste en modelar las aplicaciones

orientadas a servicios específicamente con metodologías orientadas a servicios. Surgiendo la necesidad de una metodología que guíe el desarrollo de un software orientado a servicios.

1.1. TITULO DE LA INVESTIGACION.

"INTEGRACION DE LA METODOLOGIA AGIL XP EL ESTUDIO DE WINDOWS COMMUNICATION FOUNDATION (WCF) COMO ALTERNATIVA METODOLOGICA PARA EL DESARROLLO DE SOFTWARE ORIENTADO A SERVICIOS CASO PRACTICO: CIPFIE ESPOCH."

1.2. EL PROBLEMA DE LA INVESTIGACION.

Como desarrolladores y arquitectos de software tenemos la necesidad de guiarnos con una metodología pero en algún momento nos hemos preguntado ¿Qué metodología debo usar para el desarrollo de Software?. Y de hecho esta pregunta se torna muy importante, pues como arquitectos de Software, debemos tener un plano en que apoyarnos.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no llevamos una metodología de por medio, lo que obtenemos es clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos.

El objetivo de décadas ha sido el encontrar metodologías que mejoren la productividad y la calidad del software. Seleccionar la mejor metodología para crear software es un proceso que no finaliza nunca, actualmente las metodologías orientadas a objetos constituyen el enfoque dominante en la creación de software, pero con el surgimiento de los servicios Web y las aplicaciones orientadas a servicios el uso de estas metodologías no han obtenido los resultados deseados.

Hoy en día el Centro de Investigación y Producción de la Facultad de Informática y Electrónica de la ESPOCH a pesar de la importancia que reviste como entidad encargada de la administración de los proyectos científicos no cuenta con un mecanismo informático que sea capaz de automatizar la gestión de los mismos, ya que la totalidad de la información se encuentra almacenada en archivos lógicos exponiéndola a pérdidas involuntarias e inconsistencias, además de reducir la posibilidad de tener reportes adecuados en un tiempo moderado.

Cabe mencionar que existe una tesis que señala a los Centros de Investigación de la ESPOCH como caso práctico, en la que se plantea una reingeniería y estandarización de procesos a mas de la automatización de dichas entidades a través de la implantación de un sistema software; misma que no se ha efectuado hasta la fecha, perdurando el problema del proceso manual en el cual esta inmiscuido el CIPFIE.

1.3. JUSTIFICACIÓN DE LA INVESTIGACIÓN.

La orientación a servicios es una nueva forma de modelar el desarrollo de software, considerando a los servicios como unidades de creación de software que permiten una fácil integración a los nuevos sistemas que requieran de una funcionalidad ya implementada en otros sistemas.

El modelado orientado a servicios requiere de actividades adicionales que no son encontrados en un tradicional modelado orientado a objetos, surgiendo la necesidad de una metodología que guié el desarrollo de un software orientado a servicios.

La programación Extrema (XP) es una metodología ágil para el desarrollo de software, se caracteriza principalmente por su carácter adaptativo y su orientación hacia la gente

debido a la práctica de un conjunto de valores y principios no considerados en las metodologías monumentales.

Windows Communication Foundation es un modelo para crear aplicaciones orientadas a servicios que permite a los desarrolladores que actualmente crean aplicaciones orientadas a objetos crear también aplicaciones orientadas a servicios de una manera similar. WCF conduce a los Servicios Web a un nivel superior, proporcionando a los desarrolladores una alta productividad para construir aplicaciones seguras, fiables e interoperables, simplificando sustancialmente el proceso de creación de sistemas distribuidos.

El sistema a desarrollarse constituye un elemento estratégico dentro de la gestión de los proyectos de grado regulados por el CIPFIE-ESPOCH ya que cubrirá necesidades y requerimientos enfocados en los siguientes procesos:

- Gestión de proyectos estudiantiles en cada una de sus fases (presentación, evaluación anteproyecto, seguimiento, evaluación de tesis) es decir cuando son ingresados y son considerados como anteproyectos de tesis y después de su aprobación pasando a llamarse proyectos de tesis.
- Integración con el Sistema Académico de la Institución.
- Emisión de reportes y documentos pertinentes a las bases legales del CIPFIE-ESPOCH.

Además dentro de los beneficios que el sistema brindará están los incrementos en productividad, confiabilidad, seguridad y todas las características que proporciona la automatización de lo antes mencionado.

1.4. OBJETIVOS

1.4.1. OBJETIVO GENERAL

- Integrar a la metodología ágil XP el estudio de WCF para proponer una alternativa metodológica en el desarrollo de software orientada a servicios.

1.4.2. OBJETIVOS ESPECÍFICOS

- Estudiar el paradigma arquitectónico de orientación a servicios.
- Estudiar la tecnología Windows Communication Foundation.
- Estudiar la metodología Ágil XP para determinar actividades específicas en el desarrollo de software.
- Desarrollar una metodología que guíe el desarrollo de software orientado a servicios.
- Implementar un sistema informático para el CIPFIE aplicando la metodología propuesta, mismo que estará integrado al Sistema Académico de la Institución.
- Implantar en el CIPFIE el Sistema desarrollado para beneficio de la Entidad y los Estudiantes de la Facultad de Informática y Electrónica de la ESPOCH.

1.5. HIPÓTESIS.

- La unificación de la metodología ágil XP y la tecnología WCF permitirá crear aplicaciones orientadas a servicios.

1.6. DEFINICION DE VARIABLES

VARIABLE INDEPENDIENTE: Metodología ágil XP y Tecnología WCF.

VARIABLE DEPENDIENTE: Resultados obtenidos en el desarrollo de software orientado a servicios.

CAPITULO II

ESTUDIO DEL PARADIGMA ARQUITECTÓNICO DE ORIENTACIÓN A SERVICIOS Y DE WINDOWS COMMUNICATION FOUNDATION (WCF)

INTRODUCCIÓN

Durante muchos años el principal uso de las computadoras en las organizaciones fue la automatización de las actividades individuales. El interés de las organizaciones ya no está limitado únicamente al desarrollo de software que automatice determinadas actividades individuales, sino que por el contrario, tienen como objetivo final la automatización de todo el proceso de negocio, ya que de ello depende en gran parte su competitividad.

Las necesidad de las organizaciones de poder adaptarse rápidamente a los cambios en los procesos internos que experimentan, ha motivado que se esté produciendo un

cambio de orientación que apunta hacia los procesos organizacionales o procesos de negocio. Surgen, por lo tanto, nuevas necesidades de capturar, modelar, ejecutar y monitorizar los procesos de negocio, vistos como un conjunto de procedimientos o actividades enlazadas, cuya realización permite alcanzar un cierto objetivo o meta en el contexto de una organización.

En el presente contexto el modelado de procesos y las tecnologías asociadas, tales como la Arquitectura Orientada a Servicios (SOA), ofrecen un marco adecuado para abordar el problema, puesto que cubren estas necesidades.

La propuesta de Microsoft para crear aplicaciones orientadas a servicios, cuyo nombre clave es Windows Communication Foundation (que de ahora en adelante lo llamaremos WCF), permite a los desarrolladores que actualmente crean aplicaciones orientadas a objetos con .NET Framework[1] crear también aplicaciones orientadas a servicios de una manera similar. Además, para permitir que estas aplicaciones interaccionen de manera eficaz con el software que se ejecuta en Windows y en otras plataformas, WCF implementa SOAP y otras tecnologías de servicios Web, lo que permite a los desarrolladores crear servicios confiables, seguros y transaccionales que pueden ínter operar con software ejecutado en cualquier sistema.

2.1. La Arquitectura Orientada a Servicios SOA.

La Arquitectura Orientada a Servicios (Service-Oriented Architecture, SOA) es un concepto de arquitectura de software que define la utilización de servicios como construcciones básicas para el desarrollo de aplicaciones.

[1] Componente principal de la plataforma .NET de Microsoft.

Es una arquitectura de una aplicación donde las funcionalidades se definen como servicios independientes, con interfaces invocables bien definidas, que pueden ser llamadas en secuencias dadas para formar procesos de negocios.
La Arquitectura Orientada a Servicios es un enfoque de diseño: una filosofía en armonía con los requisitos de la empresa. Requiere que usted vuelva a pensar en sus procesos empresariales y diseñe una estrategia que los exprese como segmentos reutilizables e intercambiables de la lógica empresarial.

Esto, en el fondo, significa tres cosas. En primer lugar, el desarrollo de aplicaciones puede realizarse más rápidamente. En segundo lugar, la integración se hace más fácil. Y finalmente, la reutilización de aplicaciones y procesos es una realidad, no una promesa vacía.

En otras palabras, la Arquitectura Orientada a Servicios es un diseño arquitectural que permite que los usuarios se beneficien de un conjunto de servicios sobre una red. Éstos se comunicarán unos con otros de manera que se adecuen más estrechamente a las funciones y procesos de la empresa, más que a las aplicaciones.

Una estrategia de aplicaciones empresariales debe facilitar su integración. Además que debe motivar la construcción de servicios, mas que de aplicaciones. Estos servicios se encargarían de exponer una funcionalidad bien definida a la aplicación que la requiera. De esta manera, una aplicación final simplemente orquesta la ejecución de un conjunto de estos servicios, añade su lógica particular y le presenta una interfaz al usuario final.

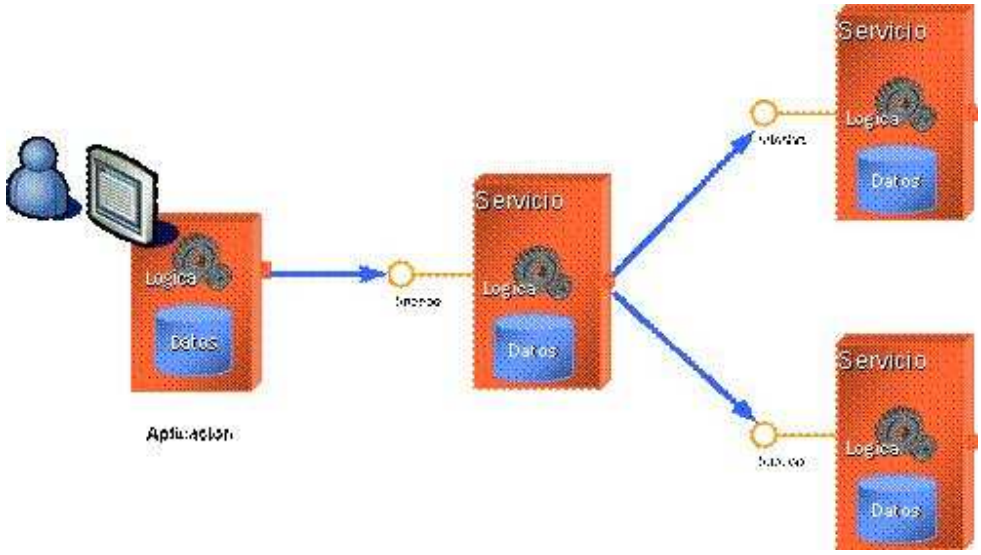


Figura II.1.- " Un sistema sencillo basado en servicios."

(Fuente.- http: //www.microsoft.com/spanish/msdnl/comunidad/mtj/voices/art143.asp, Elaborado por.- José David Parra)

Una aplicación además de implementar sus propios componentes de negocio y datos, también puede reutilizar la funcionalidad de servicios existentes en la red empresarial.

Exponer procesos de negocio como servicios es la clave de la flexibilidad de la arquitectura. Esto permite que otras piezas de funcionalidad (incluso también implementadas como servicios) hagan uso de otros servicios de manera natural, sin importar su ubicación física. Así un sistema evoluciona con la adición de nuevos servicios y su mejoramiento. Donde cada servicio evoluciona de una manera independiente. La Arquitectura Orientada a Servicios resultante, define los servicios de los cuales estará compuesto el sistema, sus interacciones, y con que tecnologías serán implementados. Las interfaces que utiliza cada servicio para exponer su funcionalidad son gobernadas por contratos, que definen claramente el conjunto de mensajes soportados, su contenido y las políticas aplicables.

2.1.1 Ventajas de una Arquitectura Orientada a Servicios

Las ventajas de la filosofía SOA son múltiples y la mayoría de ellas deriva de dos factores:

- En primer lugar, este enfoque hace posible la reutilización a gran escala del software, con lo que ello implica sobre la calidad, los costes y la facilidad de despliegue.
- En segundo, SOA independiza las aplicaciones de la infraestructura y la plataforma tecnológica.

Una de los principales habilitadores y a la vez ventajas del enfoque SOA es la posibilidad de proporcionar un acceso consistente a toda la información relevante para la empresa como un servicio, de modo que pueda ser utilizada por cualquier aplicación. Esta posibilidad eliminaría los inconvenientes que crean los datos encerrados en los típicos silos[2] de aplicación, y permitiría incorporar una mayor inteligencia a cualquier proceso de negocio.

El establecimiento de una arquitectura orientada a servicios puede ayudar a preparar tanto a IT como a los procesos de negocios para un cambio rápido. Aún en las etapas tempranas de adopción de una SOA, su organización puede beneficiarse.

- Aumente los ingresos. Cree nuevos caminos al mercado; cree nuevos valores a partir de sistemas existentes.
- Provea un modelo de negocios flexible. Reaccione más rápidamente a los cambios del mercado.
- Baje los costos. Elimine los sistemas duplicados; construya una vez y potencie;

[2] Depósitos

- Reduzca los riesgos y la exposición. Mejore la visibilidad en las operaciones de negocio.
- Disminuya los tiempos de los ciclos de desarrollo e implementación usando bloques de construcción de servicios reutilizables, prefabricados.
- Integre por toda la empresa. Incluso los sistemas históricamente separados y facilite las fusiones y adquisiciones de empresas.
- Reduzca tiempos de ciclo y costos pasando de transacciones manuales a automáticas.
- Facilite la realización de negocios con los asociados de negocios aumentando su flexibilidad.

2.1.2 Elementos esenciales de una Arquitectura Orientada a Servicios

En las Arquitecturas Orientadas a Servicios, el elemento básico es el servicio. Pero únicamente con este concepto, no podríamos diseñar una arquitectura SOA. Cuatro son los elementos esenciales necesarios para la construcción de una Arquitectura Orientada a Servicios:

1. **Operación:** Es la unidad de trabajo o procesamiento en una arquitectura SOA.
2. **Servicio:** Es un contenedor de lógica. Estará compuesto por un conjunto de operaciones, las cuales las ofrecerá a sus usuarios.
3. **Mensaje:** Para poder ejecutar una determinada operación, es necesario un conjunto de datos de entrada. A su vez, una vez ejecutada la operación, esta devolverá un resultado. Los mensajes son los encargados de encapsular esos datos de entrada y de salida.

4. **Proceso de negocio:** Son un conjunto de operaciones ejecutadas en una determinada secuencia (intercambiando mensajes entre ellas) con el objetivo de realizar una determinada tarea.

Figura II.2.- "Elementos esenciales de SOA."

(Fuente.- http: //www.arquitecturaorientadaaservicios.com/elementos-esenciales-de-una.html, Elaborado por.- Antonio Barco)

Por lo tanto, una aplicación SOA estará formada por un conjunto de procesos de negocio. A su vez esos procesos de negocio estarán compuestos por aquellos servicios que proporcionan las operaciones que se necesitan ejecutar para que el proceso de negocio llegue a buen término. Por último para ejecutar esas operaciones es necesario el envío de los datos necesarios mediante los correspondientes mensajes.

2.1.3 Visión interna de los servicios

Un servicio debe ser completamente autónomo e independiente. A pesar de esto, no es una isla, porque expone una interfaz de llamado basada en mensajes, capaz de ser accedida a través de la red.

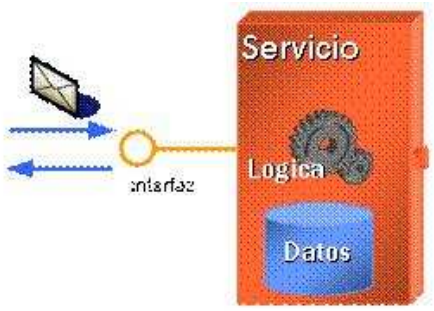


Figura II.3.- "Visión interna de los servicios."

(Fuente.- http: // www.microsoft.com/spanish/msdnl/comunidad/mtj/voices/art143.asp, Elaborado por.- José David Parra)

Un servicio funciona como una aplicación independiente. Teniendo sus propias reglas de negocio, datos, procedimientos de administración y operación. Expone toda su funcionalidad utilizando una interfaz basada en mensajes. Y por lo tanto carece de una interfaz de usuario.

La comunicación hacia y desde el servicio, es realizada utilizando mensajes y no llamadas a métodos. Estos mensajes deben contener o referenciar toda la información necesaria para entenderlo.

Un servicio es la evolución en complejidad de un componente distribuido, y se diferencian en:

- Mucho menos acoplados con sus aplicaciones cliente que los componentes.

- Menor granularidad que los componentes.
- No son diseñados e implementados necesariamente como parte de una aplicación end-to-end.
- Son controlados y administrados de manera independiente.
- Expone su funcionalidad a través de protocolos abiertos e independientes de plataforma. Incluso arriesgando el rendimiento y consumo de recursos.
- Son transparentes de su localización en la red, de esta manera garantizan escalabilidad y tolerancia a fallos.
- Tienen sus propias políticas de escalabilidad, seguridad, tolerancia a fallos, manejo de excepciones, configuración, etc.

2.1.4 Principios de la orientación a servicios

Un problema con el que nos podemos encontrar a la hora de construir una aplicación SOA es si la aplicación construida realmente es una aplicación "SOA Compliant"[3]. Para comprobar si una aplicación lo es, la mejor forma de hacerlo es chequeando que la aplicación cumpla con los Principios de la Orientación a Servicios**.**

No existe una definición estándar de cuales son los Principios de la Orientación a Servicios, por lo tanto, lo único que se puede proporcionar es un conjunto de Principios que estén muy asociados con la Orientación a Servicios.

Estos Principios según Thomas Erl son:

[3] SOA complaciente

- **Los Servicios deben ser reusables:** Todo servicio debe ser diseñado y construido pensando en su reutilización dentro de la misma aplicación, dentro del dominio de aplicaciones de la empresa o incluso dentro del dominio público para su uso masivo.

- **Los Servicios deben proporcionar un contrato formal:** Todo servicio desarrollado, debe proporcionar un contrato en el cual figuren: el nombre del servicio, su forma de acceso, las funcionales que ofrece, los datos de entrada de cada una de las funcionalidades y los datos de salida. De esta manera, todo consumidor del servicio, accederá a este mediante el contrato, logrando así la indepencia entre el consumidor y la implementación del propio servicio.

- **Los Servicios deben tener bajo acoplamiento:** Es decir, que los servicios tienen que ser independientes los unos de los otros. Para lograr ese bajo acoplamiento, lo que se hará es que cada vez que se vaya a ejecutar un servicio, se accederá a él a través del contrato, logrando así la independencia entre el servicio que se va a ejecutar y el que lo llama. Si conseguimos este bajo acoplamiento, entonces los servicios podrán ser totalmente reutilizables.

- **Los Servicios deben permitir la composición:** Todo servicio debe ser construido de tal manera que pueda ser utilizado para construir servicios genéricos de más alto nivel, el cual estará compuesto de servicios de más bajo nivel.

- **Los Servicios deben de ser autónomos:** Todo Servicio debe tener su propio entorno de ejecución. De esta manera el servicio es totalmente independiente y

nos podemos asegurar que así podrá ser reutilizable desde el punto de vista de la plataforma de ejecución.

- **Los Servicios no deben tener estado:** Un servicio no debe guardar ningún tipo de información. Esto es así porque una aplicación está formada por un conjunto de servicios, lo que implica que si un servicio almacena algún tipo de información, se pueden producir problemas de inconsistencia de datos. La solución, es que un servicio sólo contenga lógica, y que toda información esté almacenada en algún sistema de información sea del tipo que sea.

- **Los Servicios deben poder ser descubiertos:** Todo servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades. En el caso de los Servicios Web, el descubrimiento se logrará publicando los interfaces de los servicios en registros UDDI.

Cuando se desarrollan aplicaciones SOA es muy útil y necesario tener en cuenta siempre estos principios, ya que nos van a dar las pautas necesarias para tomar ciertas decisiones de diseño complejas.

Como se habrá podido observar, una característica muy importante de los **Principios de la Orientación a Servicios**, es que todos ellos se interrelacionan.

El siguiente gráfico muestra la interrelación de los diferentes principios:

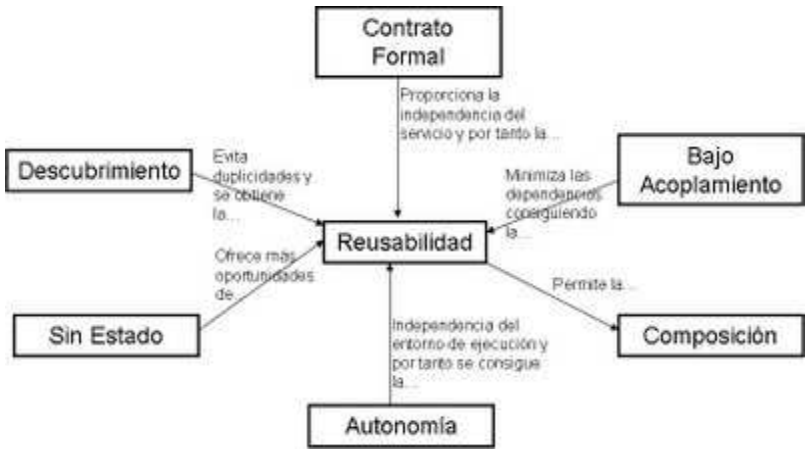


Figura II.4.- "Interrelación de los principios de la orientación a servicios."

(Fuente.- http: // www.arquitecturaorientadaaservicios.com/principios-de-la-orientacion-servicios.html, Elaborado por.- Antonio Barco)

Como se puede observar en el la Figura II.4, el objetivo de la Orientación a Servicios es obtener software totalmente reutilizable a través de un conjunto de técnicas y principios como los descritos anteriormente.

2.1.5 Modelado de Servicios

"U*na aplicación SOA está formada por un conjunto de servicios interconectados cuyo objetivo es automatizar uno o varios procesos de negocio*".

Por tanto, a la hora de construir una aplicación SOA, el elemento sobre el que debemos enfocar nuestros esfuerzos es el concepto de servicio. En este punto surgen una serie de preguntas:

- ¿Cómo puedo saber cuántos servicios se deben crear?
- ¿Qué tipos de servicios existen?

¿Cómo puedo saber cuántos servicios se deben crear? Para determinar los servicios que se deben desarrollar debemos determinar los procesos de negocio de nuestro dominio de aplicación, a partir de los cuales determinaremos cuantos y cuales servicios desarrollar.

¿Qué tipos de servicios existen? Esta pregunta se la hace todo desarrollador a la hora de enfrentarse a una aplicación SOA. Existen básicamente tres tipos de servicios, divididos en base a sus funcionalidades:

- **Servicios controladores:** Son los encargados de recibir las peticiones de los clientes y realizar las llamadas necesarias a otros servicios (en la secuencia adecuada) para devolver una respuesta. Es decir, son los servicios encargados de coordinar al resto de servicios. Si analizamos bien este tipo de servicios, nos daremos cuenta de que representan a los procesos de negocio que queremos implementar, ya que un proceso de negocio no es más que un conjunto de tareas ejecutadas en una determinada secuencia para obtener un objetivo.

- **Servicios de negocio:** Son los servicios que representan una tarea de negocio, y que forman parte de un proceso de negocio. Este tipo de servicios suelen ser poco reutilizables porque están orientados a resolver una tarea muy puntual.

- **Servicios de utilidad:** Son aquellos servicios que se caracterizan por representar una tarea altamente reutilizable. Existen dos tipos, los servicios orientados al negocio que representan una tarea de negocio altamente reutilizable entre aplicaciones y los servicios tecnológicos encargados de encapsular una

determinada tecnología y por tanto altamente reutilizables (Ej.: servicio de acceso a bases de datos relacionales).

2.1.6 Diseño SOA

La implementación ideal de un servicio exige resolver algunos inconvenientes técnicos inherentes a su modelo:

- Los tiempos de llamado no son despreciables, gracias a la comunicación de la red, tamaño de los mensajes, etc. Esto necesariamente implica la utilización de mensajería confiable.
- La respuesta del servicio es afectada directamente por aspectos externos como problemas en la red, configuración, etc. Estos deben ser tomados en cuenta en el diseño, desarrollándose los mecanismos de contingencia que eviten la parálisis de las aplicaciones y servicios que dependen de él.
- Debe manejar comunicaciones no confiables, mensajes impredecibles, reintentos, mensajes fuera de secuencia, etc.

Según lo anterior, se puede ver que la construcción de un servicio es una tarea mucho mas complicada que la de un simple componente distribuido.

El servicio debe publicar una interfaz (por ejemplo, utilizando WSDL o proxies[4]) fácilmente localizable en la red. Esta interfaz debe servir como un contrato de servicio. Donde se describen cada una de las funciones que provee, e incluso los niveles de prestación de servicio. Esta interfaz debe estar claramente documentada de manera que sea muy fácil implementar una conexión.

[4] Referencias a servicios

Un diseño exitoso de una arquitectura basada en servicios debe estar basado en una plataforma de mensajería confiable, que aísle de la implementación funcional de muchos de los problemas anteriormente mencionados. Algunas de las responsabilidades de un mecanismo así, incluyen:

- Entrega garantizada de mensajes,
- Enrutamiento de peticiones a un servicio disponible,
- Seguridad del contenido de los mensajes,
- QoS Calidad del Servicio
- Escenarios fuera de línea

Entre mas de estas características tenga la tecnología elegida, menos problemas tendrá la solución final en operación.

2.1.7 El problema de múltiples servicios

Cuando se usan múltiples servicios para implementar un sistema, es muy fácil que la comunicación entre estos se salga de control. Por ejemplo, usted puede tener un servicio que llama a otros seis servicios, algunos de los cuales llama a otros servicios, y de esta manera, muy fácilmente el sistema se vuelve inmanejable. De esta manera, un sistema grande puede terminar con múltiples dependencias.

Detectar un problema de rendimiento o funcionalidad se puede volver muy complicado.

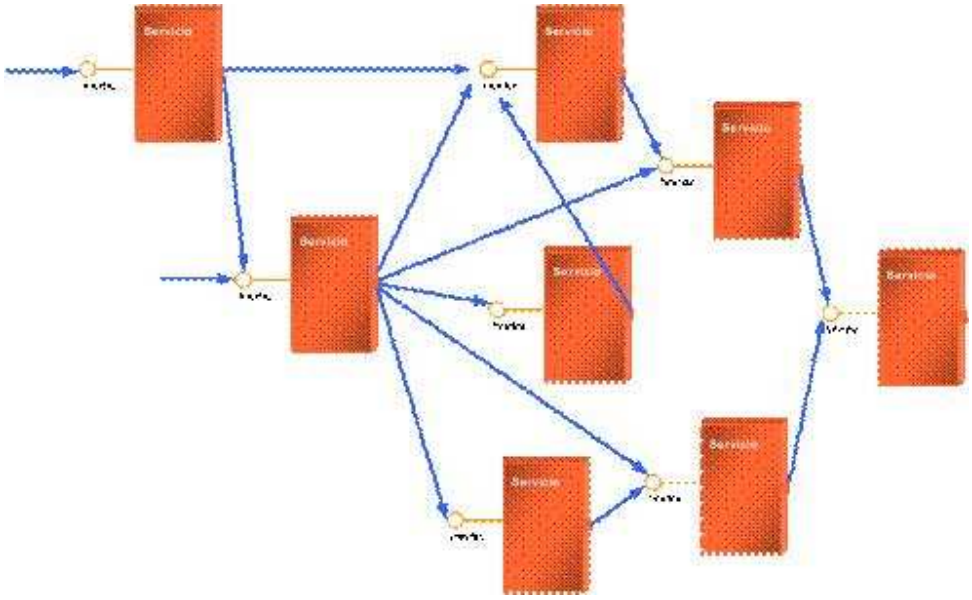


Figura II.5.- "El Problema de los multiservicios."

(Fuente.- http: //www.microsoft.com/spanish/msdnl/comunidad/mtj/voices/art143.asp, Elaborado por.- José David Parra)

Si no se cuenta con una estrategia adecuada de SOA, se puede llegar a una implementación de SOA como la anterior. Donde existe una explosión de dependencias entre los diferentes servicios.

Una solución lógica a este problema es extraer los aspectos de procedimiento de varios servicios dentro de uno dedicado, llamado servicio de negocio. Un servicio de negocio controla las acciones paso a paso en la ejecución de algún trabajo, moviendo el sistema de un estado a otro. En cada paso, este llamara una operación de negocio provista por un servicio.

Así, un servicio de negocio centraliza la definición del proceso, en vez de tener piezas del proceso entre todos los servicios. Esto disminuye las dependencias entre servicios y las aplicaciones clientes. Ayudando a facilitar la administración del sistema.

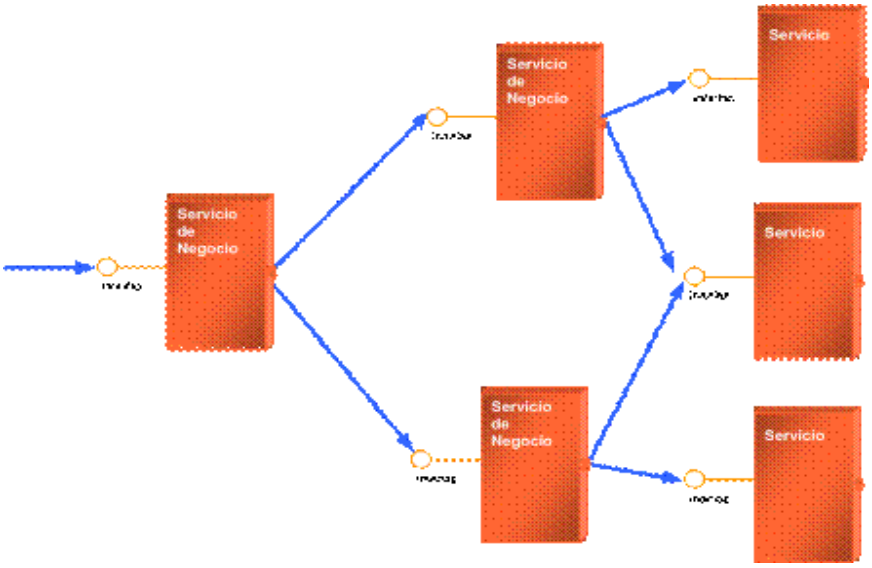


Figura II.6.- "Un modelo más simplificado."

(Fuente.- http: //www.microsoft.com/spanish/msdnl/comunidad/mtj/voices/art143.asp, Elaborado por.- José David Parra)

Introducir el concepto de servicios de negocio simplifica considerablemente la arquitectura que se mostró en la Figura II.6. Aquí simplemente consolidamos funcionalidades creando algunos servicios de negocio que hacen llamadas al resto de servicios.

2.1.8 El ciclo de vida de SOA

El ciclo de vida de SOA incluye cuatro etapas: modelaje, ensamble, despliegue y administración. El sustento de estas etapas es el control y los procesos que ofrecen orientación y control de los proyectos SOA.

2.1.8.1 Modelaje

Se empieza la etapa de modelaje reuniendo y analizando los requisitos empresariales empleados para modelar, simular y optimizar sus procesos de negocio. Se puede usar esta información para establecer una interpretación común entre el negocio y la TI de sus procesos, objetivos y resultados empresariales, además de ayudarle a asegurarse que las aplicaciones resultantes cumplan con los requisitos empresariales definidos.

2.1.8.2 Ensamble

Durante la etapa de ensamble, se crean servicios de los recursos nuevos y existentes. Cuando los nuevos servicios están disponibles, se orquestan para implantar el proceso empresarial.

2.1.8.3 Despliegue

Durante la etapa de despliegue, se configura y escala el entorno de tiempo de ejecución para cumplir con los niveles de servicio necesarios para sus procesos empresariales. Luego se pueden desplegar los procesos en un entorno de servicios robusto, escalable y altamente seguro optimizado para ejecutar de manera confiable procesos empresariales de misión crítica, al tiempo que ofrecen flexibilidad para hacer actualizaciones dinámicas en respuesta a los cambiantes requerimientos empresariales.

2.1.8.4 Administración

La etapa de administración implica establecer y mantener la disponibilidad de servicio y los tiempos de respuesta, además de manejar los activos de servicios básicos. Se pueden controlar los indicadores de desempeño relevantes tiempo real para obtener la

información necesaria para prevenir, aislar, diagnosticar y resolver problemas. Esta etapa básicamente permite al usuario tomar las mejores decisiones empresariales con mayor rapidez que antes.

2.1.9 Protocolos de mensajería

Independiente de la tecnología que se escoja para implementar la comunicación entre los servicios, esta debe manejar los problemas ya enunciados anteriormente. A continuación comparamos algunas de las opciones de protocolos a utilizar:

Tabla II.1. Protocolos de mensajería

Protocolo	**Pros:**	**Contras:**
Remoting	Tiene el mejor rendimiento si se utiliza un *serializador* binario sobre TCP.	A menos que se utilice HTTP y un *serializador* SOAP, utiliza un protocolo propietario. Aunque existen productos como Ja.NET (http://j-integra.intrinsyc.com/) que permiten interoperabilidad con Java 1.2+. No tiene las capacidades de mensajería confiable, seguridad y enrutamiento de mensajes requeridas por SOA. Deben ser implementados manualmente. Promueven una arquitectura mas acoplada.
Web Services	Protocolo abierto	Su rendimiento sobre la red no es el

XML	(independiente de plataforma) Independiente de ubicación Fácil de implementar Promueven una arquitectura desacoplada.	mejor (en comparación con un protocolo binario). Debido a que todo el flujo es texto. La serialización/deserialización en los extremos también contribuye a su pobre desempeño. Aunque cuenta con una manera de hacer llamados asincrónicos, no tiene todas las capacidades de mensajería requeridas por SOA.
Web Services XML + Web Services Enhancements (WSE) 2.0	Permite configurar un canal binario sobre el cual se va toda la mensajería SOAP. Provee muchas de las características requeridas por SOA implementando las especificaciones WS-Security, WS-Routing, WS-ReliableMessaging, entre otras.	Aunque basado en estándares, aun no existen muchas implementaciones diferentes a la de Microsoft. Esto complica la interoperabilidad. Aun falta madurez en las implementaciones actuales.
MSMQ	Por ser un protocolo de mensajería asincrónica basado en colas de mensajes, ya incluye las características que brindan confiabilidad y desacoplamiento a la	Aunque existen puentes con otros sistemas de mensajería basada en colas (como MQSeries de IBM), es un protocolo cerrado.

	transmisión de los mensajes.	

2.2 WINDOWS COMMUNICATION FOUNDATION (WCF)

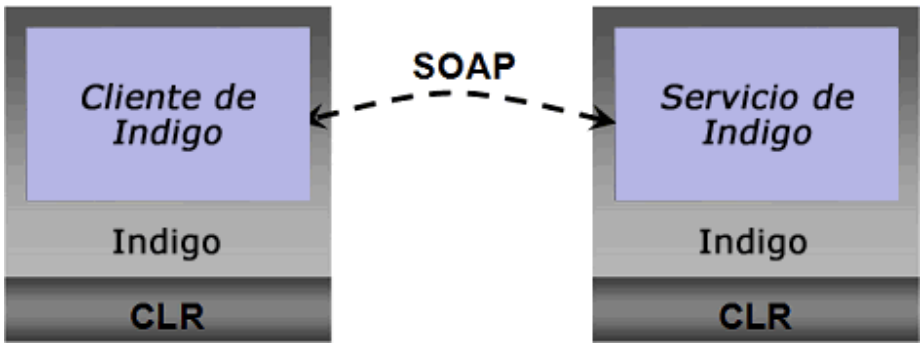


Figura II.7.- "Vista sencilla de un cliente y servicio de WCF o Indigo."

(Fuente.- http: //www.microsoft.com/spanish/msdnl/articulos/archivo/040405/voices/introindigov1.asp, Elaborado por.- David Chappell)

WCF proporciona los cimientos, implementados principalmente como un conjunto de clases que se ejecutan en Common Language Runtime (CLR), para crear los servicios a los que tendrán acceso los clientes. Un cliente y un servicio interactúan a través de SOAP, el protocolo nativo de WCF y aunque la Figura II.6 muestra ambas partes creadas con WCF, en realidad no es necesario.

2.2.1 Características de WCF

Hay tres aspectos que destacan, como las características más importantes de WCF: la unificación de varias tecnologías de Microsoft existentes, su compatibilidad con la interoperabilidad con otros proveedores y su orientación explícita a servicios.
A continuación un detalle de las mismas:

2.2.1.1 Unificación de las tecnologías de computación distribuida de Microsoft

Las primeras versiones de .NET Framework incluían varias tecnologías diferentes para crear aplicaciones distribuidas. A continuación mostraremos las tecnologías, junto con el motivo principal por el que un desarrollador utilizaría dicha tecnología.
Para crear servicios Web básicos interoperables, por ejemplo, la mejor opción eran los servicios Web ASP.NET, a los que se suelen denominar ASMX.

Para conectar dos aplicaciones basadas en .NET Framework, .NET Remoting era a veces el enfoque más adecuado. Si la aplicación requería transacciones distribuidas y servicios más avanzados, era probable que su creador utilizara Enterprise Services, el sucesor de .NET Framework a COM+. Para aprovechar las especificaciones de servicios Web más recientes, como WS-Addressing y WS-Security, un desarrollador podría crear aplicaciones que utilizaran Web Services Enhancements (WSE), la implementación inicial de Microsoft de estas nuevas especificaciones.

Por último, para crear aplicaciones con colas y basadas en mensajes, un desarrollador basado en Windows podría utilizar Microsoft Message Queuing (MSMQ).

Aunque todas estas opciones eran interesantes, esta diversidad resultaba bastante desconcertante para los desarrolladores. ¿Por qué disponer de tantas opciones?.

Una solución mejor consistía en tener una única tecnología que resolviese todos estos problemas, como ha ocurrido con la llegada de WCF. En vez de obligar a los desarrolladores a elegir una de varias posibilidades, WCF les permite crear aplicaciones distribuidas que resuelven todos los problemas de los que antes se encargaban las tecnologías que subsume.

Aunque Microsoft seguirá ofreciendo compatibilidad con las tecnologías anteriores, la mayoría de las nuevas aplicaciones que antes hubieran utilizado estas tecnologías ahora se crearán en WCF.

2.2.1.2 Interoperabilidad con las aplicaciones que no son de Microsoft

Conseguir que la vida de los desarrolladores sea más sencilla gracias a la unificación de tecnologías diversas es una buena idea. Pero con el acuerdo universal sobre servicios Web entre los proveedores, también es posible resolver el eterno problema de la interoperabilidad entre aplicaciones. Al ser SOAP el mecanismo fundamental de comunicación de WCF, las aplicaciones de WCF pueden comunicarse con software que se ejecuta en una gran variedad de contextos. Tal como vemos en la siguiente figura, una aplicación creada en WCF puede interactuar con todas las aplicaciones siguientes:

- Aplicaciones de WCF ejecutadas en un proceso diferente del mismo equipo de Windows.
- Aplicaciones de WCF ejecutadas en otro equipo de Windows.
- Aplicaciones creadas con otras tecnologías, como servidores de aplicaciones basados en Java 2, Enterprise Edition (J2EE), que admitan servicios Web estándar.

Estas aplicaciones se pueden ejecutar en equipos de Windows o en equipos con otros sistemas operativos, como Sun Solaris, z/OS de IBM o Linux.

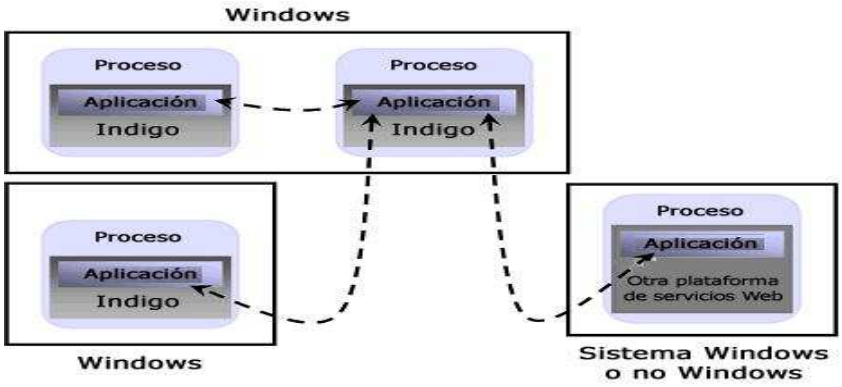


Figura II.7- "Interoperabilidad de WCF o Indigo con otras aplicaciones que no son de Microsoft."

(Fuente.- http://www.microsoft.com/spanish/msdnl/articulos/archivo/040405/voices/introindigov1.asp, Elaborado por.- David Chappell)

Las aplicaciones de WCF también pueden ínter operar con aplicaciones creadas en algunas de las tecnologías de .NET Framework anteriores a WCF, por ejemplo ASMX, como se describe más adelante.

Para permitir una comunicación más amplia que la básica, WCF implementa un grupo de novedosas tecnologías de servicios Web a las que se denomina colectivamente como especificaciones WS-*. Estos documentos definen métodos de varios proveedores para agregar transacciones, seguridad, mensajería confiable, etc. a servicios Web basados en SOAP. Todas estas especificaciones fueron definidas en un principio por Microsoft, IBM y otros proveedores mediante un trabajo conjunto. Tras estabilizarse, la propiedad suele transferirse a un organismo de normalización como OASIS (Organización para el desarrollo de estándares de la información estructurada).

Cuando una aplicación de WCF se comunica con una aplicación que se ejecuta en un sistema que no es Windows, se utiliza el protocolo estándar SOAP (tal vez con algunas extensiones WS-*), representadas en la conexión mediante la habitual codificación de XML basado en texto. Cuando una aplicación basada en WCF se comunica con otra

aplicación basada en WCF, no obstante, es conveniente optimizar esta comunicación. Aunque se ofrecen todas las mismas características, incluidas las transacciones, seguridad y mensajería confiable, la codificación de la conexión es una versión binaria optimizada de SOAP. Los mensajes siguen la estructura de datos de un mensaje SOAP, conocido como *Infoset*[5], pero la codificación utiliza una representación binaria del Infoset en vez de los habituales corchetes angulares y el formato de texto del XML.

2.2.1.3 Compatibilidad explícita con el desarrollo Orientado a Servicios

La idea de considerar una aplicación que proporciona y consume servicios no es muy nueva. La novedad consiste en centrarse directamente en los servicios y diferenciarlos de los objetos. Con este propósito, los creadores de WCF han tenido muy en cuenta cuatro principios fundamentales a la hora de diseñar esta tecnología

- Compartir el esquema, no la clase: a diferencia de las anteriores tecnologías de objetos distribuidos, los servicios interactúan con los clientes sólo mediante una interfaz XML bien definida. No se permiten algunos comportamientos como la transferencia de clases completas, métodos y todo, a través de los límites de los servicios.

- Los servicios son autónomos: un servicio y sus clientes acuerdan la interfaz existente entre ellos, pero aparte de eso son independientes. Por tanto, se pueden escribir en diferentes lenguajes, utilizar diferentes entornos de ejecución, como CLR y la máquina virtual Java, ejecutarse en diferentes sistemas operativos y diferir en otros aspectos.

[5] Conjunto de datos

- Los límites son explícitos: un objetivo de las tecnologías de objetos distribuidos como COM distribuido (DCOM) era conseguir que los objetos remotos fueran lo más similares posible a los objetos locales. Aunque este enfoque simplificaba el desarrollo en cierto modo al proporcionar un modelo de programación común, también ocultaba las diferencias inevitables entre objetos locales y objetos remotos. Los servicios evitan este problema al hacer más explícitas las interacciones entre los servicios y sus clientes. No se intenta ocultar la distribución.

- Utilizar compatibilidad basada en directivas: siempre que sea posible, la especificación de las opciones que se utilizarán entre sistemas debe fundamentarse en mecanismos basados en directivas WS-Policy.

La orientación a servicios es un campo amplio, que abarca las aplicaciones orientadas a servicios y los conceptos, más generales, de la arquitectura orientada a servicios (SOA). WCF constituirá los cimientos de aplicaciones orientadas a servicios creadas en Windows y, por lo tanto, será fundamental para los trabajos realizados en SOA de muchas organizaciones.

2.2.2 EL modelo de Programación de WCF

2.2.2.1 Creación de un servicio de WCF

Como muestra la Figura II.8, todos los servicios de WCF se crean mediante tres elementos:

- Una *clase de servicio*, implementada en C#, en VB.NET o en otro lenguaje basado en CLR, que implementa uno o varios métodos;
- Un entorno *host*, un proceso y un dominio de aplicación, en el que se ejecuta el servicio;

- Uno o varios *extremos* que permiten a los clientes tener acceso al servicio.

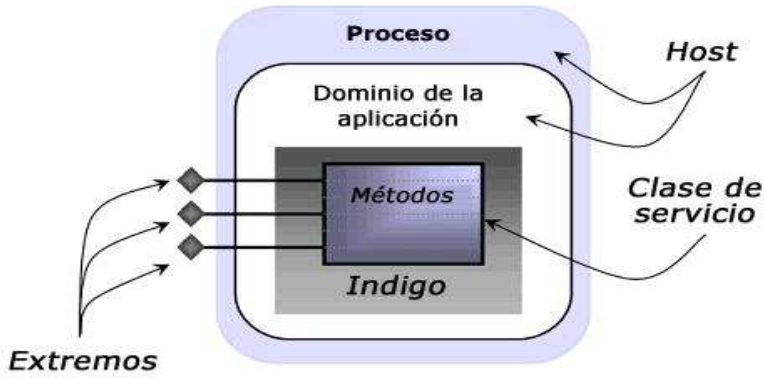


Figura II.8- "Un servicio de WCF o Indigo."

(Fuente.- http://www.microsoft.com/spanish/msdnl/articulos/archivo/040405/voices/introindigov1.asp, Elaborado por.- David Chappell)

Toda la comunicación con un servicio de WCF se realiza a través de los extremos del servicio. Cada extremo especifica un *contrato* que identifica los métodos a los que se puede tener acceso mediante este extremo, un *enlace* que determina cómo puede comunicarse un cliente con este extremo y una *dirección* que indica dónde se puede encontrar este extremo.

Para entender WCF es necesario comprender bien todos estos conceptos:

2.2.2.2 Creación de una clase de servicio

Una clase de servicio de WCF es una clase como cualquier otra, pero con algunas novedades, que permiten al creador de la clase definir uno o varios *contratos* que la clase implementa. Cada servicio de WCF implementa al menos un *contrato de servicio*, que define las operaciones que el servicio expone. La clase de servicio también puede implementar explícitamente un *contrato de datos*, que define los datos que transmiten estas operaciones.

2.2.2.3 Definición de contratos de servicio

Todas las clases de servicio de WCF implementan los métodos que pueden utilizar los clientes. El creador de la clase de servicio determina los métodos que se exponen como operaciones a las que el cliente puede llamar mediante su inclusión en un contrato de servicio. La definición de los contratos de servicio, de hecho, el trabajar explícitamente con los servicios en general, es en gran medida una idea novedosa para el mundo .NET. Los creadores de WCF necesitaban encontrar una manera de aplicar esta idea al CLR y los lenguajes de programación que incluye. Afortunadamente, los creadores del CLR previeron la necesidad de este tipo de extensiones, por lo que incorporaron la posibilidad de utilizar *atributos*. Desde el punto de vista del desarrollador, los atributos son cadenas de caracteres, tal vez con propiedades asociadas, que pueden aparecer antes de una definición de clase, una definición de método y otros lugares. Dondequiera que aparece un atributo, cambia ciertos aspectos del comportamiento del elemento al que está asociado.

.NET Framework ha utilizado los atributos con distintos objetivos desde su lanzamiento inicial. Por ejemplo, para marcar que un método es un servicio Web al que se puede llamar mediante SOAP en la tecnología ASMX de Framework, dicho método se precede con el atributo WebMethod. De manera similar, Enterprise Services utiliza el atributo Transacción para indicar que un método requiere una transacción. WCF aplica esta idea a los servicios y establece una gama de atributos que permiten definir y controlar los servicios.

El atributo más importante de WCF es ServiceContract. De hecho, una clase de servicio de WCF es sencillamente una clase que se marca con el atributo ServiceContract o que

implementa una interfaz marcada con este atributo. A continuación, podemos ver un sencillo ejemplo en C# que utiliza el primer método:

```
using System.ServiceModel;
[ServiceContract]
class Calculadora
{
 [OperationContract]
 private int Suma (int a, int b)
 {
   return a + b;
 }
 [OperationContract]
 public int Resta (int a, int b)
 {
   return a - b;
 }
    public int Multiplicacion (int a, int b)
 {
    return a * b;
 }
}
```

El atributo ServiceContract y todos los demás atributos que WCF utiliza se definen en el espacio de nombres System.ServiceModel, por lo que el ejemplo comienza con una instrucción using que hace referencia a este espacio de nombres. Todos los métodos

de una clase de servicio a los que el cliente puede llamar deben marcarse con otro atributo denominado OperationContract. WCF expone todos los métodos de una clase de servicio que aparecen precedidos por el atributo OperationContract como operaciones a las que se puede llamar mediante SOAP. En este ejemplo, tanto Suma como Resta están marcados con este atributo por lo que quedan expuestos a los clientes de este servicio. Todos los métodos de una clase de servicio que no estén marcados con OperationContract, como Multiplicacion en el ejemplo anterior, no se incluirán en el contrato de servicio, por lo que los clientes de este servicio de WCF no podrán llamarlos.

Dos abstracciones bastante diferentes, servicios y objetos, se unen en WCF. Es importante comprender que ambas se basan en contratos, de manera explícita o implícita, para definir lo que exponen al mundo exterior. Un objeto, especificado por alguna clase, define de forma efectiva un contrato que determina cuáles son los métodos a los que puede llamar otro objeto de la misma aplicación. El acceso a estos métodos queda controlado por palabras clave del lenguaje como public y private. En la clase Calculadora mostrada anteriormente, por ejemplo, otros objetos de la misma aplicación pueden llamar a Resta y Multiplicacion, que son los dos métodos públicos de la clase. El contrato de objetos de esta clase expone únicamente estos dos métodos.

Mediante los atributos de WCF, Calculadora también define un contrato de servicio, como acabamos de ver. Este contrato también tiene dos métodos, pero no coinciden con los del contrato de objetos. El atributo OperationContract controla si un cliente del servicio de WCF puede llamar a un determinado método, en vez de las palabras public y private. Como este atributo aparece únicamente en Add y Resta, los clientes sólo pueden llamar a estos dos métodos. Los contratos de objetos y los contratos de

servicios son completamente independientes y por tanto un método como Add puede ser private aunque lleve el atributo OperationContract.

El ejemplo que acabamos de ver muestra la manera más sencilla de crear una clase de servicio de WCF: basta con marcar una clase directamente con ServiceContract.

Una vez hecho esto, el contrato de servicio de la clase se define implícitamente y consta de todos los métodos de la clase marcados con OperationContract. También es posible (y probablemente mejor en la mayoría de los casos) especificar los contratos de servicio de manera explícita mediante un tipo interface del lenguaje. Empleando este método, la clase Calculadora podría tener el siguiente aspecto:

```
using System.ServiceModel;
[ServiceContract]
public interface ICalculadora
{
 [OperationContract]
 private int Suma (int a, int b);
 [OperationContract]
 public int Resta (int a, int b);
}

class Calculadora : ICalculadora
{
 private int Suma (int a, int b)
 {
```

```
  return a + b;
 }
 public int Resta (int a, int b)
 {
  return a - b;
 }
public int Multiplicacion (int a, int b)
 {
   return a * b;
  }
}
```

En este ejemplo, los atributos ServiceContract y OperationContract se asignan a la interfaz ICalculadora y los métodos que contiene en vez de a la clase Calculadora propiamente dicha. El resultado es el mismo, no obstante, de manera que esta versión del servicio expone el mismo contrato de servicio que la anterior. El uso de interfaces explícitas como la anterior es ligeramente más complicado, pero también ofrece una mayor flexibilidad. Por ejemplo, una clase puede implementar más de una interfaz, lo que permite implementar por tanto más de un contrato de servicio. Al exponer varios extremos, cada uno de ellos con un contrato de servicio diferente, una clase puede presentar diferentes grupos de servicios para diferentes clientes.

Un último detalle: al marcar una clase de servicio con ServiceContract y OperationContract también es posible generar automáticamente definiciones de contrato de servicio en Lenguaje de descripción de servicios Web (WSDL). Por lo tanto, se puede obtener acceso a la definición visible externamente de cada contrato de servicio de WCF como un documento WSDL estándar que especifica las operaciones del

contrato. Y aunque no se describe aquí, también es posible crear una clase de servicio de WCF directamente a partir de un documento WSDL, un método que resulta especialmente útil para implementar interfaces WSDL definidas externamente.

2.2.2.4 Definición de contratos de datos

Una clase de servicio de WCF especifica un contrato de servicio que define los métodos que se exponen a los clientes del servicio. Todas estas operaciones también suelen implicar ciertos datos, lo que significa que un contrato de servicio también implica cierto contrato de datos que describa la información que se intercambiará. En algunos casos, este contrato de datos se define implícitamente como parte del contrato de servicio. Por ejemplo, en las clases Calculadora mostradas anteriormente, cada método utiliza dos parámetros, ambos enteros, y devuelve un único entero.
Estos parámetros definen todos los datos que intercambia este servicio, por lo que constituyen el contrato de datos del servicio. Para servicios como este, en el que cada operación utiliza únicamente tipos simples, tiene sentido definir los aspectos de los datos del contrato de manera implícita en el contrato de servicio. No es necesario nada más.

Pero los servicios también pueden tener parámetros de tipos más complejos, como estructuras. En este tipo de casos, es necesario un contrato de datos explícito. Los contratos de datos definen cómo se convierten los tipos en memoria a un formato apto para su transmisión a través del cable, proceso denominado serialización. De hecho, los contratos de datos son un mecanismo para controlar cómo se serializan los datos.

En una clase de servicio de WCF, se define un contrato de datos utilizando el atributo DataContract. Una clase, estructura u otro tipo marcado con DataContract puede tener uno o varios de sus miembros precedido por el atributo DataMember, lo que indica que este miembro debe incluirse en un valor serializado de este tipo. Veamos un sencillo ejemplo:

```
[DataContract]
struct Cliente {
 [DataMember] public string Nombre;
 int public edad;
 [DataMember] private int Cedula;
}
```

Cuando una instancia de este tipo Cliente se pasa como parámetro en un método marcado con OperationContract, sólo se pasarán los campos marcados con el atributo DataMember (Nombre y Cedula).

El hecho de que un campo esté etiquetado como public o private no tiene ningún efecto sobre cómo se serializará el campo. Al igual que ocurre con los métodos, las palabras clave public y private son parte del contrato que define cómo otros objetos de la misma aplicación pueden tener acceso a este tipo. DataMember, al igual que OperationContract, define cómo los clientes del servicio que implementa esta clase pueden tener acceso al tipo. Nuevamente, los dos son completamente diferentes.

Un último punto que merece la pena señalar acerca de los contratos de WCF es que de manera predeterminada no hay nada que forme parte de un contrato de servicio o de un contrato de datos. Por el contrario, un desarrollador debe utilizar explícitamente los

atributos ServiceContract y DataContract para indicar los tipos que tienen contratos definidos por WCF y, a continuación, especificar explícitamente las partes de dichos tipos que se exponen a los clientes de este servicio utilizando los atributos OperationContract y DataMember. Uno de los principios básicos de los diseñadores era que los servicios tuvieran límites explícitos y, por tanto, WCF es una tecnología que permite elegir. Todo lo que un servicio pone a disposición de sus clientes deberá especificarse expresamente en el código.

Los contratos y los atributos que los definen son uno de los aspectos más importantes de WCF y esta breve descripción sólo trata los puntos más destacados. El atributo OperationContract se puede utilizar para definir operaciones *unidireccionales*, por ejemplo, si una llamada a un servicio no tiene respuesta. También es posible definir interacciones en las que ambas partes pueden actuar como cliente y servicio, cada una de ellas invocando operaciones y exponiendo las operaciones invocadas por la otra parte, mediante la creación de contratos llamados *dúplex*. El atributo DataContract ofrece varias opciones más, siendo posible incluso trabajar directamente con mensajes SOAP utilizando un atributo llamado MessageContract. Los contratos se utilizan para expresar muchos de las posibilidades que ofrece WCF, por lo que constituyen uno de sus conceptos más importantes.

2.2.2.5 Selección de un host

Una clase que implementa un servicio de WCF suele compilarse en una biblioteca. Por definición, todas las bibliotecas necesitan un dominio de aplicación host y un proceso de Windows en el que se ejecuten. WCF ofrece dos opciones para alojar bibliotecas que implementen servicios. Una opción consiste en utilizar el dominio de aplicación host y el proceso que ofrece el servicio de activación de Windows (WAS, Windows Activation

Service), mientras que la otra permite que un servicio se aloje en cualquier dominio de aplicación que se ejecute en cualquier proceso. Esta sección describe ambas, comenzando por WAS.

2.2.2.5.1 Alojamiento de un servicio mediante el servicio de activación de Windows

La manera más sencilla de alojar un servicio de WCF es utilizar WAS. El uso de WAS es muy similar al mecanismo de alojamiento que IIS ofrece para ASMX. Entre otros puntos en común, ambos se basan en el concepto de *directorio virtual*, que no es más que un alias más corto de una ruta de directorio real del sistema de archivos de Windows.

Para ver cómo funciona el alojamiento en WAS, supongamos que alguna de las clases Calculadora antes mostradas se compila en una biblioteca llamada calc.dll y, a continuación, se coloca en el directorio virtual *Calculadora* de un sistema que ejecuta Windows Server 2003. Para indicar que el servicio de WCF implementado en calc.dll debe alojarse en WAS, un desarrollador crea un archivo en el directorio virtual Calculadora con la extensión .svc (que corresponde, por supuesto, a "servicio"). Para nuestro sencillo ejemplo, este archivo podría llamarse calc.svc y sólo contendría lo siguiente:

```
%@Service language=c# class="Calculadora" %
```

Una vez hecho esto y definido un extremo como se explica en la siguiente sección, una solicitud de un cliente a uno de los métodos del servicio Calculadora creará automáticamente una instancia de esta clase para que ejecute la operación

especificada. Dicha instancia se ejecutará en un dominio de aplicación creado dentro del proceso estándar que ofrece WAS.

2.2.2.5.2 Alojamiento de un servicio en un proceso cualquiera

Aunque utilizar WAS para que proporcione un proceso para alojar un servicio de WCF es la opción más sencilla, con frecuencia las aplicaciones necesitan exponer servicios desde su propio proceso en vez de utilizar el que ofrece Windows. Afortunadamente, no es difícil hacerlo. El siguiente ejemplo muestra cómo crear un proceso que aloje cualquier de las clases Calculadora definidas anteriormente:

```
using System.ServiceModel;
public class CalculadoraHost
{
 public static void Main()
 {
  ServiceHost<Calculadora> s1 =
    new ServiceHost<Calculadora>();
  s1.Open();
  Console.Writeline("Press ENTER to end service");
  Console.Readline();
 }
}
```

Como la clase CalculadoraHost incluye un método Main, se ejecutará como un proceso diferente. Para alojar el servicio de ejemplo Calculadora, este método debe crear una nueva instancia de la clase ServiceHost<T>, pasando la clase Calculadora. (Observe

que esta clase estándar de WCF es un *componente genérico*, como indican los < y > que rodean a su parámetro. Los componentes genéricos son una nueva característica del lenguaje en la versión 2.0 de C#, Visual Basic .NET y otros lenguajes basados en la versión 2.0 de .NET Framework.) Una vez que se crea una instancia de esta clase, para que el servicio esté disponible basta con llamar al método Open de dicha instancia. WCF dirigirá automáticamente las solicitudes de los clientes a los métodos apropiados de la clase Calculadora.

Para permitir que un servicio de WCF pueda procesar las solicitudes de sus clientes, el proceso que lo aloja debe seguir ejecutándose, lo que no supone un problema con los servicios alojados por WAS, ya que el proceso estándar WAS garantiza esto. No obstante, una aplicación host debe resolver este problema por sí misma. En este sencillo ejemplo, se mantiene ejecutando el proceso mediante el sencillo mecanismo de esperar una entrada por parte de un usuario de la consola.

2.2.2.6 Definición de extremos

Junto con la definición de las operaciones de una clase de servicio de WCF y la especificación de un proceso host que ejecute dichas operaciones, un servicio de WCF debe exponer uno o varios extremos. Todos los extremos especifican los tres elementos siguientes:

- Un nombre de *contrato* que indica el contrato de servicio que esta clase de servicio de WCF expone mediante este extremo.

Una clase marcada con ServiceContract que no implementa ninguna interfaz explícita, como Calculadora en el primer ejemplo mostrado anteriormente, sólo puede exponer

un contrato de servicio. En este caso, todos sus extremos expondrán el mismo contrato. Si una clase implementa explícitamente dos o más interfaces marcadas con ServiceContract, no obstante, extremos diferentes pueden exponer contratos diferentes.

Una *dirección* que indica dónde se puede encontrar este extremo. Las direcciones son direcciones URL que identifican a un equipo y un extremo concreto de dicho equipo.

Un *enlace* que determina cómo se puede tener acceso a ese extremo. El enlace determina la combinación de protocolos que se puede utilizar para tener acceso a ese extremo justo con otros datos, como si la comunicación es confiable y los mecanismos de seguridad que es posible utilizar. Imaginemos, por ejemplo, que el creador de un servicio desea permitir a los clientes que obtengan acceso al servicio mediante SOAP a través de HTTP o SOAP a través de TCP. Al ser cada uno de estos un enlace diferente y, por tanto, el servicio debería exponer dos extremos, uno con un enlace de SOAP a través de HTTP y el otro con un enlace de SOAP a través de -TCP.

Los enlaces son una parte fundamental de cómo se lleva a cabo la comunicación. Para facilitar su uso, WCF incluye un conjunto de enlaces predefinidos, cada uno de los cuales especifica un determinado grupo de opciones. Este conjunto incluye:

BasicProfileHttpBinding: cumple el perfil básico 1.0 de la organización Web services Interoperability (WS-I), que especifica SOAP a través de HTTP. Este es el enlace predeterminado del enlace si no se especifica otro explícitamente.

- BasicProfileHttpsBinding: cumple el perfil 1.0 de seguridad básico de WS-I, que especifica el SOAP a través de HTTPS.

- WsHttpBinding: admite transferencia confiable de mensajes con WS-ReliableMessaging, seguridad con WS-Security y transacciones con WS-AtomicTransaction. Este enlace permite la interoperabilidad con otras implementaciones de servicios Web que también admiten estas especificaciones.
- WsDualHttpBinding: igual que WsHttpBinding, pero también admite la interacción mediante contratos dúplex. Mediante este enlace, tanto servicios como clientes pueden recibir y enviar mensajes.
- NetTcpBinding: envía SOAP con codificación binaria, incluida la compatibilidad con la transferencia confiable de mensajes, seguridad y transacciones, directamente a través de TCP. Este enlace sólo se puede utilizar para la comunicación de WCF a WCF.
- NetNamedPipeBinding: envía SOAP con codificación binaria a través de canalizaciones con nombre. Este enlace sólo se puede utilizar para la comunicación de WCF a WCF entre procesos que se encuentren en el mismo equipo de Windows.
- NetMsmqBinding: envía SOAP con codificación binaria a través de MSMQ, como se describe anteriormente. Este enlace sólo se puede utilizar para la comunicación de WCF a WCF.

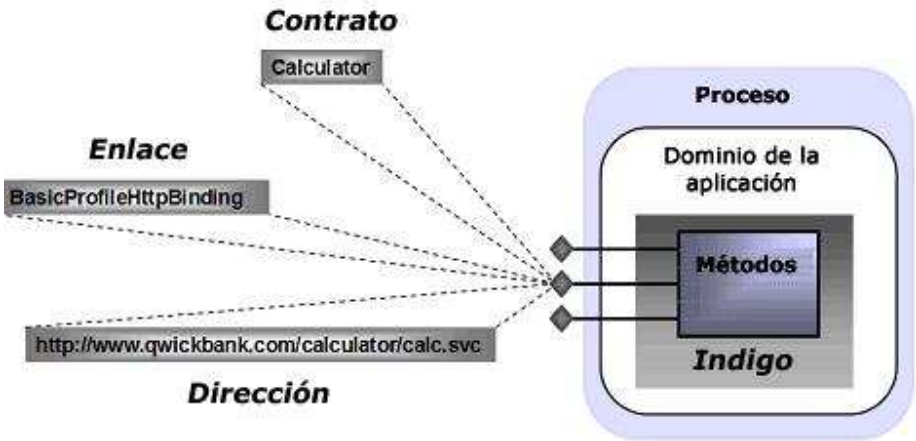


Figura II.9- "Elementos de un extremo o endpoind de WCF o Indigo."

(Fuente.- http://www.microsoft.com/spanish/msdnl/articulos/archivo/040405/voices/introindigov1.asp, Elaborado por.- David Chappell)

La Figura II.9 muestra valores de ejemplo para cada uno de los tres elementos de un extremo para el primer servicio Calculadora que se mostraba anteriormente.

El nombre del contrato de servicio es Calculadora, que es el nombre de la clase que implementa este servicio, y el enlace es BasicProfileHttpBinding. Si suponemos que este servicio se aloja mediante WAS, se ha instalado en el directorio virtual *Calculadora* descrito anteriormente y se ejecuta en un equipo llamado bank.com, la dirección podría ser http://www.bank.com/Calculadora/calc.svc.

A diferencia de los contratos, los extremos no se definen utilizando atributos. Aunque es posible crear extremos mediante programación, el método más habitual probablemente consiste en utilizar un archivo de configuración asociado al servicio. Los servicios alojados en WAS utilizan el archivo web.config, mientras que los servicios alojados de manera independiente utilizan el archivo de configuración asociado a la aplicación en la que se ejecutan (al que suele hacerse referencia como app.config, aunque el nombre de archivo real puede variar). Si se utiliza para la primera clase de servicio Calculadora mostrada anteriormente, este archivo de configuración podría tener el siguiente aspecto:

```
<configuration>
 <system.serviceModel>
  <services>
   <service serviceType="Calculadora">
```

```
        <endpoint
        contractType="Calculadora"
        bindingType="basicProfileHttpBinding />
    </service>
  </services>
 </system.serviceModel>
</configuration>
```

El elemento system.serviceModel contiene la información de configuración de todos los servicios que implementa una aplicación de WCF. Este elemento contiene un elemento services que puede contener uno o varios elementos service. Este sencillo ejemplo sólo tiene un único servicio, por lo que service sólo aparece una vez. El atributo serviceType del elemento service identifica la clase de servicio que implementa el servicio al que se aplica esta configuración, que en este caso es Calculadora. Cada elemento service puede contener uno o varios elementos endpoint, cada uno de los cuales especifica un determinado extremo mediante el que se puede obtener acceso a este servicio de WCF. En este ejemplo, el servicio sólo expone un único extremo y, por tanto, sólo aparece un único elemento endpoint. El nombre del contrato de este extremo es Calculadora, que es el nombre de la clase que lo implementa. Si este archivo de configuración fuese para la segunda clase de servicio Calculadora mostrada anteriormente, que definía el contrato de servicio mediante una interfaz explícita, el valor del atributo serviceType seguiría siendo el mismo, pero el valor de contractType pasaría a ser ICalculadora, que es el nombre de esta interfaz explícita. El enlace especificado aquí es basicProfileHttpBinding, aunque al ser el predeterminado, se podría haber omitido. Si suponemos además que Calculadora es un servicio alojado en

WAS, se crearía una dirección automáticamente, por lo que no sería necesario especificar una en este archivo de configuración.

2.2.2.7 Creación de un Cliente de WCF

Si crear un servicio básico de WCF no es muy complicado, crear un cliente de WCF es todavía más sencillo. Basta con crear un emplazamiento[6] local para dicho servicio, llamado *proxy*, que se conectará a un extremo concreto en el servicio de destino y, a continuación, invocar las operaciones del servicio a través de este proxy.

La Figura II.10 muestra el aspecto de todo esto.

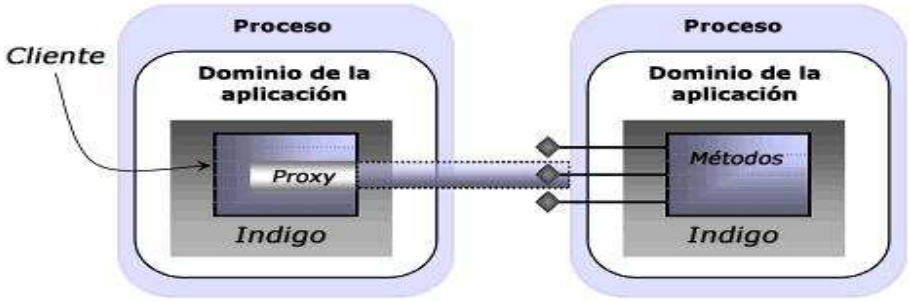


Figura II.10- "Cliente de WCF o Indigo."

(Fuente.- http://www.microsoft.com/spanish/msdnl/articulos/archivo/040405/voices/introindigov1.asp, Elaborado por.- David Chappell)

Para crear un proxy es necesario saber exactamente el contrato que expondrá el extremo de destino y, a continuación, utilizar esta definición del contrato para generar el proxy. En WCF, este proceso se realiza mediante una herramienta denominada svcutil. Si el servicio se implementa utilizando WCF, svcutil puede obtener acceso a la DLL del servicio para detectar el contrato y generar un proxy.

Si sólo está disponible la definición WSDL del servicio, svcutil puede leerla para generar un proxy. Si sólo el propio servicio está disponible, svcutil puede obtener

[6] Instalación, localización

acceso a él directamente mediante WS-MetadataExchange o un sencillo HTTP GET para adquirir la definición de interfaz de WSDL del servicio y, a continuación, generar el proxy.

Independientemente de cómo se genere, el cliente puede crear una nueva instancia del proxy y, a continuación, invocar los métodos del servicio mediante dicha instancia. A continuación, podemos ver un sencillo ejemplo de un cliente para la clase Calculadora:

```
using System.ServiceModel;
using WCF.Example; // namespace for generated proxy class
public class CalculadoraClient
{
  public static void Main()
  {
    CalculadoraProxy p = new CalculadoraProxy();
    Console.WriteLine("7 + 2 = {0}", p.Add(7, 2));
    Console.WriteLine("7 - 2 = {0}", p.Resta(7, 2));
    p.Close();
  }
}
```

2.2.3 Otros aspectos de WCF

Los conceptos básicos de los servicios y los clientes son esenciales para todas las aplicaciones de WCF. Sin embargo, la mayoría de estas aplicaciones utilizarán también otros aspectos de esta tecnología.

2.2.3.1 Control del comportamiento local

Muchos aspectos de WCF, como los contratos, los enlaces, etc. están relacionados con la comunicación entre un servicio y sus clientes. Sin embargo, hay otras partes del comportamiento de un servicio que son básicamente locales. Por ejemplo, cómo se controla la duración de la instancia de un servicio y cómo se administra el acceso simultáneo a dicha instancia. Para permitir que los desarrolladores puedan controlar comportamientos como estos, WCF define dos atributos principales, ambos con varias propiedades. Uno de estos atributos, ServiceBehavior, se puede aplicar a las clases que también están marcadas con el atributo ServiceContract. El otro, OperationBehavior, se puede aplicar a los métodos de una clase de servicio que también está marcada con el atributo OperationContract.

El atributo ServiceBehavior tiene varias propiedades que afectan al comportamiento del servicio de manera global. Por ejemplo, se puede utilizar una propiedad denominada ConcurrencyMode para controlar el acceso simultáneo al servicio. Si se establece en Single, WCF entregará únicamente una solicitud de cliente a este servicio a la vez, es decir, el servicio tendrá un único subproceso. Si se establece en Multiple, WCF entregará más de una solicitud de cliente a la vez al servicio, cada uno de ellos ejecutándose en un subproceso diferente. De manera similar, la propiedad

InstanceMode de ServiceBehavior se puede utilizar para controlar cómo se crean y destruyen las instancias de un servicio. Si InstanceMode se establece en PerCall, se creará una nueva instancia del servicio para tratar cada solicitud de cliente, la cual se destruirá cuando finalice la solicitud. Si se establece en PrivateSession, no obstante, la misma instancia del servicio se utilizará para tratar todas las solicitudes procedentes de un determinado cliente.

Imaginemos, por ejemplo, que su creador ha decidido que la clase Calculadora debe tener varios subprocesos y utilizar la misma instancia para todas las llamadas de un determinado cliente. La definición de la clase tendría el siguiente aspecto:

```
using System.ServiceModel;
[ServiceContract]
[ServiceBehavior(
  ConcurrencyMode=Multiple,
  InstanceMode=PrivateSession)]
class Calculadora { ... }
```

De manera similar, las propiedades del atributo OperationBehavior permiten controlar el comportamiento de suplantación del método que implementa esta operación, sus requisitos transaccionales (descritos más adelante) y otros detalles.

2.2.3.2 Opciones de mensajería

Los sencillos ejemplos que se muestran en este artículo suponen que se utiliza un método de llamada a procedimiento remoto (RPC) sincrónico para la interacción entre cliente y servicio. WCF admite esta opción, pero no es la única posibilidad.

SOAP es un protocolo orientado a mensajes, lo que implica que puede admitir varios modelos de programación. De hecho, WCF admite varias posibilidades, entre ellas:

- RPC tradicional, que bloquea las llamadas que incluyen listas de parámetros con tipo;
- RPC asincrónica, que no bloquea las llamadas que incluyen listas de parámetros con tipo;
- Mensajería tradicional, que no bloquea las llamadas que incluyen un único parámetro de mensaje;
- RPC basada en mensajes, que bloquea las llamadas que incluyen un único parámetro de mensaje.

Además, aunque es necesario para la mayoría de las aplicaciones distribuidas, la especificación SOAP no trata en ningún momento la confiabilidad. Una manera habitual de garantizar la confiabilidad consiste en utilizar SOAP únicamente en escenarios de punto a punto y emplear TCP para garantizar la entrega de solicitudes y respuestas, lo cual es suficiente en algunos casos y es lo que se hace cuando se utiliza BasicProfileHttpBinding.

Sin embargo, también hay muchos casos en los que no es suficiente. Por ejemplo, ¿qué ocurre si se obtiene acceso a un servicio mediante varios intermediarios SOAP? Las garantías de confiabilidad que ofrece TCP no son suficientes en este caso para garantizar la confiabilidad de un extremo a otro. Para resolver este problema, WCF implementa la especificación WS-ReliableMessaging. Si se elige un enlace como

WsHttpBinding, que utiliza WS-ReliableMessaging, un servicio y su cliente pueden garantizar una comunicación confiable de un extremo a otro aunque sea a través de varios intermediarios SOAP.

2.2.3.3 Seguridad

La exposición de servicios en una red, aunque sea una red interna, normalmente requiere ciertas medidas de seguridad. ¿Cómo puede el servicio comprobar la identidad del cliente? ¿Cómo se pueden proteger los mensajes enviados al servicio o desde él de alteraciones malintencionadas y ojos indiscretos? ¿Y cómo se puede limitar el acceso a un servicio para que únicamente los usuarios autorizados puedan utilizarlo? Sin alguna solución a estos problemas, resulta demasiado peligroso exponer numerosos tipos de servicios. Sin embargo, la creación de aplicaciones seguras puede resultar complicada. Lo ideal sería que hubiese maneras directas de tratar escenarios habituales de seguridad, junto con un control más preciso para las aplicaciones que lo requieran.

Para lograrlo, WCF ofrece funciones de seguridad básicas de autenticación, integridad de los mensajes, confidencialidad de los mensajes y autorización. El método de WCF para tratar los primeros tres aspectos se basa principalmente en los enlaces y entre las opciones del desarrollador se encuentran las siguientes:

- Seleccionar un enlace estándar que admita seguridad. Las aplicaciones que necesiten sólo seguridad basada en el transporte, por ejemplo, pueden utilizar un enlace como BasicProfileHttpsBinding. Este método es suficiente para solicitudes que vayan directamente de un cliente a un servicio sin atravesar ningún

intermediario, como servidores proxy HTTP u otros nodos SOAP. Las aplicaciones que requieren seguridad de un extremo a otro para mensajes que atraviesan varios intermediarios SOAP pueden utilizar en su lugar un enlace que admita seguridad WS-Security, como WsHttpBinding.

- Seleccionar un enlace estándar que admita seguridad y, a continuación, personalizarlo cambiando uno o varios de los valores predeterminados. Por ejemplo, el mecanismo de autenticación que utiliza un enlace como WsHttpBinding se puede cambiar si se desea.

- Crear un enlace personalizado que ofrezca exactamente las características que un desarrollador necesita. Aunque este proceso no es apto para cardiacos, puede ser la solución adecuada para ciertos escenarios avanzados.

- Seleccionar un enlace estándar que no admita seguridad, como BasicProfileHttpBinding. Aunque no aplicar ninguna seguridad es una práctica bastante peligrosa en general, es posible que sea la mejor opción en ciertas situaciones.

Un servicio de WCF también puede controlar los clientes autorizados a utilizar sus servicios. En general, WCF admite únicamente los mecanismos de autorización que incluye .NET Framework. Un servicio puede utilizar el atributo estándar PrincipalPermission, por ejemplo, para definir quién tiene permiso para obtener acceso a él.

Permitir a los desarrolladores que creen aplicaciones seguras sin exponerles a una complejidad desorbitada ha sido todo un reto en el pasado. Al proporcionar un método directo para los casos más habituales y un control preciso para situaciones más complejas, WCF intenta lograr este objetivo de una manera viable y eficaz.

2.2.3.4 Coexistencia y Migración

WCF constituye un método moderno de crear aplicaciones distribuidas en la era de los servicios confiables, seguros y transaccionales. Un punto clave que hay que señalar, no obstante, es que la instalación de WCF no impedirá el funcionamiento de las aplicaciones existentes. El código actual que se ejecuta en ASMX, .NET Remoting y las demás tecnologías cuya funcionalidad subsume WCF continuarán funcionando, por lo que no será necesario cambiar a WCF. Pero aquellas organizaciones que hayan invertido en las tecnologías actuales de Microsoft, queda una pregunta evidente: ¿qué ocurrirá con el código existente escrito utilizando las tecnologías que han precedido a WCF?

Para todas las tecnologías actuales cuyo futuro se verá gravemente afectado por la llegada de WCF, los desarrolladores necesitan saber dos cosas: si las aplicaciones creadas en esta tecnología interoperarán con aplicaciones creadas en WCF y cuánto trabajo implicará convertir las aplicaciones de esta tecnología al entorno de WCF. A continuación, veremos brevemente cómo cada tecnología trata estos problemas:

- Servicios Web ASP.NET (ASMX): los servicios Web creados con ASMX interoperarán con las aplicaciones de WCF. Como tanto los servicios Web ASP.NET como WCF admiten SOAP estándar, no resulta sorprendente. El cambio del código de los

servicios Web ASP.NET a WCF requerirá cierto trabajo mecánico, pero debe ser bastante directo. La estructura básica de las dos tecnologías es muy similar, por lo que en general bastará con cambiar únicamente los atributos y los archivos de configuración. Otras características más avanzadas, no obstante, como las extensiones SOAP, no se podrán pasar directamente a WCF. En su lugar, será necesario reescribirlas utilizando las opciones de extensibilidad que ofrece WCF.

- .NET Remoting: las aplicaciones creadas en .NET Remoting no interoperarán con las aplicaciones creadas en WCF, ya que sus protocolos de conexión no son compatibles. El paso del código de .NET Remoting existente a WCF requerirá cierto esfuerzo, pero será posible. Aquellos que hayan creado extensiones de .NET Remoting personalizadas, no obstante, como canales y receptores, descubrirán que este código no se puede convertir al nuevo mundo. Aunque hay extensiones similares en WCF, las interfaces que emplean no coinciden con las de .NET Remoting.

- Enterprise Services: para permitir que una aplicación existente de Enterprise Services pueda interoperar con los clientes de WCF (u otro software basado en servicios Web), un desarrollador puede identificar exactamente las interfaces que se deben exponer en dicha aplicación. Utilizando una herramienta incluida con WCF, es posible crear automáticamente contratos de servicio para dichas interfaces y exponerlos mediante WCF. Para los clientes existentes de las aplicaciones de Enterprise Services que no se hayan creado en .NET Framework (y para otros clientes basados completamente en COM), se proporciona un moniker de WCF para permitir el acceso directo a los servicios Web. El esfuerzo necesario para convertir las aplicaciones existentes de Enterprise Services y que se ejecuten directamente

en WCF será similar al necesario para pasar aplicaciones ASMX. Gran parte del trabajo, aunque no todo, serán cambios mecánicos directos a los atributos y espacios de nombres.

- Web services Enhancements (WSE): WSE es la solución táctica de Microsoft para la implementación de aplicaciones de servicios Web que requieran todas o parte de las funciones que ofrecen las especificaciones WS-*. Las aplicaciones creadas con WSE 1.0 y WSE 2.0 no interoperarán con las aplicaciones creadas en WCF.

- MSMQ: como las funciones de colas de WCF se basan en MSMQ, las aplicaciones de colas creadas en WCF pueden interoperar con las aplicaciones de colas creadas directamente en MSMQ. La conversión de aplicaciones del espacio de nombres System.Messaging incluidas con el .NET Framework original requerirá cierto trabajo, ya que esta interfaz anterior es diferente de lo que proporciona WCF. Una vez que se lance WCF, los desarrolladores deberán utilizarla en vez de System.Messaging para crear la mayoría de las aplicaciones de colas basadas en MSMQ.

La presentación de un nuevo software siempre afecta a lo que ya existe. Al ofrecer unos cimientos comunes para la creación de aplicaciones orientadas a servicios, WCF ofrece una plataforma más sencilla y coherente a los desarrolladores. Aunque este cambio provocará algunos quebraderos de cabeza, el objetivo de los creadores de WCF es conseguir que esta transición sea lo más suave y sencilla posible.

WCF representa una importante evolución respecto al modo de crear software por parte de los desarrolladores. A medida que las aplicaciones orientadas a servicios sean

más y más habituales, WCF se convertirá en una tecnología de uso general para los desarrolladores de software de Windows. WCF será la base de gran parte de la comunicación de Windows ya que proporciona los cimientos estándar para el software orientado a servicios.

El impacto que tendrá esta tecnología no será pequeño. Todos los que creen aplicaciones distribuidas en Windows, especialmente aplicaciones que deban interoperar con las de otras plataformas, deberán estar muy atentos WCF cambiará considerablemente su mundo.

CAPITULO III

ESTUDIO DE LA METODOLOGÍA ÁGIL XP

INTRODUCCIÓN

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería. Las metodologías ingenieriles han estado presentes durante mucho tiempo. No se han distinguido precisamente por ser muy exitosas. Aún menos por su popularidad. La crítica más frecuente a estas metodologías es que son burocráticas. Hay tanto que hacer para seguir la metodología que el ritmo entero del desarrollo se retarda.

Como una reacción a estas metodologías, un nuevo grupo de metodologías ha surgido en los últimos años. Durante algún tiempo se conocían como metodologías ligeras, pero el término aceptado ahora es metodologías ágiles. Para mucha gente el encanto de estas metodologías ágiles es su reacción ante la burocracia de las metodologías monumentales.

Estos nuevos métodos buscan un justo medio entre ningún proceso y demasiado proceso, proporcionando simplemente suficiente proceso para que el esfuerzo valga la pena.

El resultado de todo esto es que los métodos ágiles cambian significativamente algunos de los énfasis de los métodos ingenieriles. La diferencia inmediata es que son menos orientados al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada. De muchas maneras son más bien orientados al código: siguiendo un camino que dice que la parte importante de la documentación es el código fuente.

La falta de documentación en las metodologías ágiles es un síntoma de diferencias mucho más profundas:

- **Los métodos ágiles son adaptables en lugar de predictivos**. Los métodos ingenieriles tienden a intentar planear una parte grande del proceso del software en gran detalle para un plazo largo de tiempo, esto funciona bien hasta que las cosas cambian. Así que su naturaleza es resistirse al cambio. Para los métodos

ágiles, no obstante, el cambio es bienvenido. Intentan ser procesos que se adaptan y crecen en el cambio, incluso al punto de cambiarse ellos mismos.

- **Los métodos ágiles son orientados a la gente y no orientados al proceso.** La meta de los métodos ingenieriles es definir un proceso que funcionará bien con cualquiera que lo use. Los métodos ágiles afirman que ningún proceso podrá nunca maquillar las habilidades del equipo de desarrollo, de modo que el papel del proceso es apoyar al equipo de desarrollo en su trabajo. Explícitamente puntualizan el trabajar a favor de la naturaleza humana en lugar de en su contra y enfatizan que el desarrollo de software debe ser una actividad agradable.

3.1 Metodología Extreme Programming (XP)

La programación extrema es una metodología reciente en el desarrollo de software. La filosofía de XP es satisfacer al completo las necesidades del cliente, por eso lo integra como una parte más del equipo de desarrollo.

XP fue inicialmente creada para el desarrollo de aplicaciones dónde el cliente no sabe muy bien lo que quiere, lo que provoca un cambio constante en los requisitos que debe cumplir la aplicación. Por este motivo es necesaria una metodología ágil como XP que se adapta a las necesidades del cliente y dónde la aplicación se va reevaluando en periodos de tiempo cortos.

XP está diseñada para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, dónde la comunicación sea más factible que en grupos de

desarrollo grandes. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes.

3.2 Valores de la metodología XP

Esta metodología promueve los siguientes valores:

3.2.1 Comunicación

El Extreme Programming se nutre del ancho de banda más grande que se puede obtener cuando existe algún tipo de comunicación: la comunicación directa entre personas. Es muy importante entender cuales son las ventajas de este medio.

Cuando dos (o más) personas se comunican directamente pueden no solo consumir las palabras formuladas por la otra persona, sino que también aprecian los gestos, miradas, etc. que hace su compañero. Sin embargo, en una conversación mediante el correo electrónico, hay muchos factores que hacen de esta una comunicación, por así decirlo, mucho menos efectiva.

3.2.2 Coraje

El coraje es un valor muy importante dentro de la programación extrema. Un miembro de un equipo de desarrollo extremo debe de tener el coraje de exponer sus dudas, miedos, experiencias sin "embellecer" éstas de ninguna de las maneras. Esto es muy importante ya que un equipo de desarrollo extremo se basa en la confianza para con sus miembros. Faltar a esta confianza es una falta más que grave.

3.2.3 Simplicidad

Dado que no se puede predecir como va a ser en el futuro, el software que se esta desarrollando; un equipo de programación extrema intenta mantener el software lo más sencillo posible. Esto quiere decir que no se va a invertir ningún esfuerzo en hacer un desarrollo que en un futuro pueda llegar a tener valor. En el XP frases como "...en un futuro vamos a necesitar..." o "Haz un sistema genérico de..." no tienen ningún sentido ya que no aportan ningún valor en el momento.

3.2.4 Feedback

La agilidad se define (entre otras cosas) por la capacidad de respuesta ante los cambios que se van haciendo necesarios a lo largo del camino. Por este motivo uno de los valores que nos hace más ágiles es el continuo seguimiento o feedback que recibimos a la hora de desarrollar en un entorno ágil de desarrollo. Este feedback se toma del cliente, de los miembros del equipo, en cuestión de todo el entorno en el que se mueve un equipo de desarrollo ágil.

3.3 Prácticas de XP

3.3.1 Planificación incremental

La Programación Extrema asume que la planificación nunca será perfecta y que variará en función de como varíen las necesidades del negocio. Por tanto, el valor real reside en obtener rápidamente un plan inicial, y contar con mecanismos de feedback que permitan conocer con precisión dónde estamos. Como es lógico, la planificación es iterativa: un representante del negocio decide al comienzo de cada iteración qué características concretas se van a implementar.

El objetivo de la XP es generar versiones de la aplicación tan pequeñas como sea posible, pero que proporcionen un valor adicional claro, desde el punto de vista del negocio. A estas versiones se las denomina releases.

Una release cuenta con un cierto número de historias. La historia es la unidad de funcionalidad en un proyecto XP, y corresponde a la mínima funcionalidad posible que tiene valor desde el punto de vista del negocio. Durante cada iteración se cierran varias historias, lo que hace que toda iteración añada un valor tangible para el cliente.

Es fundamental en toda esta planificación la presencia de un representante del cliente, que forma parte del equipo y que decide cuáles son las historias más valiosas. Estas historias son las que se desarrollarán en la iteración actual.

La obtención de feedback que permita llevar a cabo estimaciones precisas es fundamental. Se hacen estimaciones para cada historia, de modo que en cuanto se comienzan a tener datos históricos, éstos se utilizan para hacer que las siguientes estimaciones sean más precisas.

Como se puede ver, y como siempre ocurre con la Programación Extrema, el enfoque utilizado para llevar a cabo la planificación es eminentemente pragmático.

Gran parte de la eficacia de este modelo de planificación deriva de una división clara de responsabilidades, que tiene en cuenta las necesidades del negocio en todo momento. Dentro de esta división, el representante del cliente tiene las siguientes responsabilidades:

- Decidir qué se implementa en cada release o iteración

- Fijar las fechas de fin de la release, recortando unas características o añadiendo otras.
- Priorizar el orden de implementación, en función del valor de negocio.

Las responsabilidades del equipo de desarrollo son las siguientes:

- Estimar cuánto tiempo llevará una historia: este feedback es fundamental para el cliente, y puede llevarle a reconsiderar qué historias se deben incluir en una iteración.

Proporcionar información sobre el coste de utilizar distintas opciones tecnológicas.

- Organizar el equipo.
- Estimar el riesgo de cada historia.
- Decidir el orden de desarrollo de historias dentro de la iteración.

3.3.2 Testing

La ejecución automatizada de tests es un elemento clave de la XP. Existen tantos tests internos (o tests de unidad), para garantizar que el mismo es correcto, como tests de aceptación, para garantizar que el código hace lo que debe hacer. El cliente es el responsable de definir los tests de aceptación, no necesariamente de implementarlos. Él es la persona mejor calificada para decidir cuál es la funcionalidad más valiosa.

El hecho de que los tests sean automatizados es el único modo de garantizar que todo funciona: desde el punto de vista de la XP, si no hay tests, las cosas sólo funcionan en apariencia. Aún más, si un test no está automatizado, no se le puede considerar como tal.

El objetivo de los tests no es corregir errores, sino prevenirlos. Por ejemplo, los tests siempre se escriben antes que el código a testear, no después: esto aporta un gran valor adicional, pues fuerza a los desarrolladores a pensar cómo se va a usar el código que escriben, poniéndolos en la posición de consumidores del software.

Elaborar los tests exige pensar por adelantado cuáles son los problemas más graves que se pueden presentar, y cuáles son los puntos dudosos. Esto evita muchos problemas y dudas, en lugar de dejar que aparezcan "sobre la marcha".

Un efecto lateral importante de los tests es que dan una gran seguridad a los desarrolladores: es posible llegar a hacer cambios más o menos importantes sin miedo a problemas inesperados, dado que proporcionan una red de seguridad. La existencia de tests hace el código muy maleable.

3.3.3 Programación en parejas

La XP incluye, como una de sus prácticas estándar, la programación en parejas. Nadie programa en solitario, siempre hay dos personas delante del ordenador. Ésta es una de las características que más se cuestiona al comienzo de la adopción de la XP dentro de un equipo, pero en la práctica se acepta rápidamente y de forma entusiasta.

El principal argumento contra la programación en parejas es que es improductiva. Esta idea se basa en el hecho de que dos programadores "programan el doble por separado". Esto sería así si no fuera por las siguientes razones:

- El hecho de que todas las decisiones las tomen al menos dos personas proporciona un mecanismo de seguridad enormemente valioso.

- Con dos personas responsabilizándose del código en cada momento, es menos probable que se caiga en la tentación de dejar de escribir tests, etc., algo fundamental para mantener el código en buena forma. Es muy difícil que dos personas se salten tareas por descuido o negligencia.
- El hecho de programar en parejas permite la dispersión de conocimiento por todo el equipo. Este efecto es difícil de conseguir de otro modo, y hace que la incorporación de nuevos miembros al equipo sea mucho más rápida y eficaz.
- El código siempre está siendo revisado por otra persona. La revisión de código es el método más eficaz de conseguir código de calidad, algo corroborado por numerosos estudios, muchos de los cuáles son anteriores a la Programación Extrema.
- En contra de lo que pueda parecer, los dos desarrolladores no hacen lo mismo: mientras el que tiene el teclado adopta un rol más táctico, el otro adopta un rol más estratégico, preguntándose constantemente si lo que se está haciendo tiene sentido desde un punto de vista global.
- Los datos indican que la programación en parejas es realmente más eficiente. Si bien se sacrifica un poco de velocidad al comienzo, luego se obtiene una velocidad de crucero muy superior. Esto contrasta con lo que ocurre en la mayor parte de los proyectos, en los que se arranca con una velocidad enorme pero rápidamente se llega a un estado muy parecido a la parálisis, en el que progresos cada vez más pequeños consumen cantidades de tiempo cada vez más grandes. Todos conocemos proyectos que se pasan el 50% del tiempo en el estado de "finalizado al 90%".

3.3.4 Refactorización

A la hora de la verdad, el código de la mayor parte de las aplicaciones empieza en un razonable buen estado, para luego deteriorarse de forma progresiva. El coste desorbitado del mantenimiento, modificación y ampliación de aplicaciones ya existente se debe en gran parte a este hecho.

Uno de los objetivos de la XP es mantener la curva de costes tan plana como sea posible, por lo que existen una serie de mecanismos destinados a mantener el código en buen estado, modificándolo activamente para que conserve claridad y sencillez. A este proceso básico para mantener el código en buena forma se le llama refactorización.

La refactorización no sólo sirve para mantener el código legible y sencillo: también se utiliza cuando resulta conveniente modificar código existente para hacer más fácil implementar nueva funcionalidad.

3.3.5 Diseño simple

Otra práctica fundamental de la Programación Extrema es utilizar diseños tan simples como sea posible. El principio es "utilizar el diseño más sencillo que consiga que todo funcione". Se evita diseñar características extra porque a la hora de la verdad la experiencia indica que raramente se puede anticipar qué necesidades se convertirán en reales y cuáles no. La XP nos pide que no vivamos bajo la ilusión de que un diseño puede resolver todas o gran parte de las situaciones futuras: lo que parece necesario cambia con frecuencia, es difícil acertar a priori.

Es obvio que, si no vamos a anticipar futuras necesidades, debemos poder modificar el diseño si alguna de estas se materializa. La XP soporta estas modificaciones gracias a los tests automatizados. Estos permiten hacer cambios importantes gracias a la red de protección que proporcionan. La refactorización, que hace que el código existente sea claro y sencillo, también ayuda a hacer factibles las modificaciones.

La XP define un "diseño tan simple como sea posible" como aquél que:

- Pasa todos los tests.
- No contiene código duplicado.
- Deja clara la intención de los programadores (enfatiza el qué, no el cómo) en cada línea de código.
- Contiene el menor número posible de clases y métodos.

3.3.6 Propiedad colectiva del código

La XP aboga por la propiedad colectiva del código. En otras palabras, todo el mundo tiene autoridad para hacer cambios a cualquier código, y es responsable de ellos. Esto permite no tener que estar esperando a otros cuando todo lo que hace falta es algún pequeño cambio.

Por supuesto, cada cuál es responsable de las modificaciones que haga. El principio básico es "tú lo rompes, tú lo arreglas, no importa si está en el código propio o en el de otros".

Por último, vale la pena tener en cuenta que la existencia de tests automatizados impide que se produzca un desarrollo anárquico, al ser cada persona responsable de

que todos los tests se ejecuten con éxito al incorporar los cambios que ha introducido al programa.

3.3.7 Integración Continua

En muchos casos la integración de código produce efectos laterales imprevistos, y en ocasiones la integración puede llegar a ser realmente traumática, cuando dejan de funcionar cosas por motivos desconocidos. La Programación Extrema hace que la integración sea permanente, con lo que todos los problemas se manifiestan de forma inmediata, en lugar de durante una fase de integración más o menos remota.

La existencia de una fase de integración separada tiene dos efectos laterales indeseables: se empieza a hacer codificación "yo-yo", en la que todo el mundo modifica código "sólo para que funcione, ya lo ajustaremos", y hace que se acumulen defectos. Evitar que se acumulen defectos es muy importante para la XP, como lo es el conseguir que los defectos que cada programador inyecta los elimine él mismo.

3.3.8 Cliente en el equipo

Algunos de los problemas más graves en el desarrollo son los que se originan cuando el equipo de desarrollo toma decisiones de negocio críticas. Esto no debería ocurrir, pero a la hora de la verdad con frecuencia no se obtiene feedback del cliente con la fluidez necesaria: el resultado es que se ha de optar por detener el avance de los proyectos, o por que el desarrollo tome una decisión de negocio. Por otra parte, los representantes del negocio también suelen encontrarse con problemas inesperados debido a que tampoco reciben el feedback adecuado por parte de los desarrolladores.

La XP intenta resolver este tipo de problemas integrando un representante del negocio dentro del equipo de desarrollo. Ésta persona siempre está disponible para resolver dudas y para decidir qué y qué no se hace en cada momento, en función de los intereses del negocio. Debido a su inmersión dentro del equipo, y a que es él quien decide qué y qué no se hace, junto con los tests que verifican si la funcionalidad es la correcta y deseada, esta persona obtiene un feedback absolutamente realista del estado del proyecto.

3.3.9 Releases pequeñas

Siguiendo la política de la XP de dar el máximo valor posible en cada momento, se intenta liberar nuevas versiones de las aplicaciones con frecuencia. Éstas deben ser tan pequeñas como sea posible, aunque deben añadir suficiente valor como para que resulten valiosas para el cliente.

3.3.10 Semanas de 40 horas

La Programación Extrema lleva a un modo de trabajo en que el equipo está siempre al 100%. Una semana de 40 horas en las que se dedica la mayor parte del tiempo a tareas que suponen un avance puede dar mucho de sí, y hace innecesario recurrir a sobreesfuerzos excepto en casos extremos.

Además, el sobreesfuerzo continuado pronto lleva a un rendimiento menor y a un deterioro de la moral de todo el equipo.

3.3.11 Estándares de codificación

Para conseguir que el código se encuentre en buen estado y que cualquier persona del equipo pueda modificar cualquier parte del código es imprescindible que el estilo de codificación sea consistente. Un estándar de codificación es necesario para soportar otras prácticas de la XP.

Sin embargo, la XP también es pragmática en esto, y apuesta por establecer un número mínimo de reglas: el resto se irán pactando de facto. Esto evita un ejercicio inicial más o menos estéril.

3.3.12 Uso de Metáforas

La comunicación fluida es uno de los valores más importantes de la Programación Extrema: la programación en parejas, el hecho de incorporar al equipo una persona que represente los intereses del negocio y otras prácticas son valiosas entre otras cosas porque potencian enormemente la comunicación.

Para conseguir que la comunicación sea fluida es imprescindible, entre otras cosas, utilizar el vocabulario del negocio. También es fundamental huir de definiciones abstractas. Dicho de otro modo, la XP no pretende seguir la letra de la ley, sino su espíritu. Dentro de este enfoque es fundamental buscar continuamente metáforas que comuniquen intenciones y resulten descriptivas, enfatizando el qué por delante del cómo.

3.4 Roles en XP

Los roles de acuerdo con la propuesta original de Beck son:

- **Programador**. El programador escribe las pruebas unitarias y produce el código del sistema.
- **Cliente**. Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- **Encargado de pruebas (Tester).** Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- **Encargado de seguimiento (Tracker)**. Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- **Entrenador (Coach).** Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- **Consultor.** Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- **Gestor (Big boss).** Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

3.5 Características de la metodología XP

- **Los individuos e interacciones son más importantes que los procesos y herramientas.**

  Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

- **Software que funcione es más importante que documentación exhaustiva.**

  Desarrollar software que funciona más que conseguir una buena documentación. La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental.

- **La colaboración con el cliente es más importante que la negociación de contratos.**

  La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

- **La respuesta ante el cambio es más importante que el seguimiento de un plan.**

  Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

3.6 Ciclo de vida de un proyecto XP

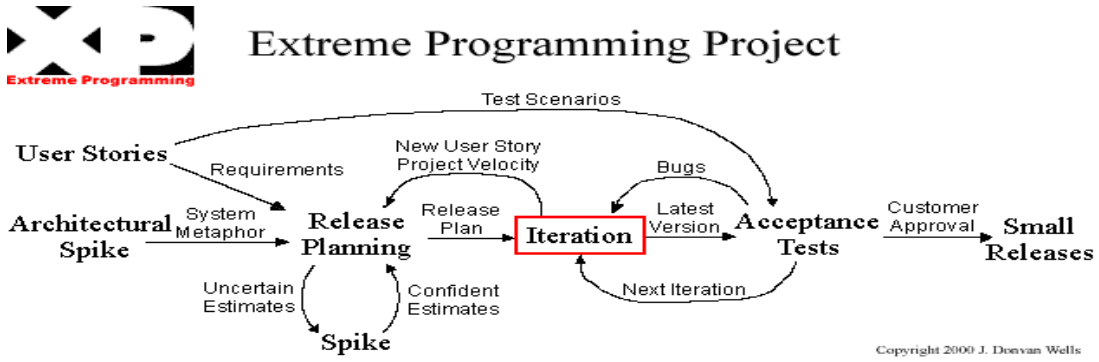


Figura III.1- "Ciclo de vida XP."

(Fuente.- http://www.oness.sourceforge.net/proyecto/html/ch05s02.html, Elaborado por.- J Deavan Wells)

El ciclo de vida ideal de XP consiste de seis fases:

3.6.1 Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto.

Las historias de usuario tienen la misma finalidad que los casos de uso pero con algunas diferencias: Constan de 3 ó 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados, etc. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen.

También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia. El tiempo de desarrollo ideal para una historia de usuario es entre 1 y 3 semanas.

Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

3.6.2 Planificación de la Entrega (*Release*)

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana

ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

3.6.3 Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en

tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

3.6.4 Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

3.6.5 Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

3.6.6 Muerte del Proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también

ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

CAPITULO V

CASO PRACTICO: DESARROLLO DEL SISTEMA INFORMÁTICO DE GESTIÓN DE PROYECTOS DE GRADO DEL CIPFIE-ESPOCH.

5.1 FASE 1: DEFINICIÓN DEL ÁMBITO

5.1.1 EQUIPO DE TRABAJO

Director:

Ing. Danilo Pastor

Miembros:

Ing. Gloria Arcos

Ing. Ivan Menes

Sr. Juan Pablo Tixe

Sr. Luís Rivera

Colaboradores:

Lcda. Maria Agusta Larrea

5.1.2 ANTECEDENTES HISTÓRICOS DE LA ORGANIZACIÓN

La Facultad de Informática y Electrónica de la Escuela Superior Politécnica de Chimborazo considerando que la investigación científico tecnológica es factor básico del desarrollo económico, social y cultural de un país, creó el Centro de Investigaciones y Producción (CIPFIE) como unidad académica administrativa, que se encargará de la planificación, ejecución, asesoramiento, coordinación y control de los proyectos de Investigación, Tesis, Memorias y se rige por el presente reglamento.

5.1.2.1 OBJETIVOS

Objetivo General

- Planificar, ejecutar, asesorar, coordinar y controlar desarrollo de proyectos de Investigación, Tesis, Memorias y Prácticas de Pre-grado, regido en el reglamento interno del Centro de Investigación de la Facultad.

Objetivos Específicos

- Planificar y coordinar planes y programas de investigación básica y aplicada, en el marco de los lineamientos de investigación y producción de la institución.
- Coordinar las actividades relacionadas con las Tesis, Memorias y Prácticas de Pre-grado.
- Promover y contribuir al desarrollo de nuevas tecnologías.
- Contribuir al mejoramiento de la enseñanza aprendizaje con las ciencias:

  De conformidad con el pénsum de estudios de las escuelas existentes, a través

De la coordinación de proyectos de investigación acorde con la realidad nacional.

- Controlar, evaluar y difundir los resultados de la investigación y producción.

5.1.2.2 SERVICIOS

- Desarrollar infraestructura investigativa, Realizar cursos de especialización o actualización, seminarios, mesas redondas, respecto al desarrollo de investigación.
- Difundir periódicamente los resultados de las investigaciones a través de revistas y otros medios tecnológicos de comunicación.
- Buscar y tramitar el auspicio a través de CONUEP, FUNDACYT, CONIN y otros organismos nacionales e internacionales para el desarrollo de los proyectos de investigación que este centro considere de importancia para su ejecución.

5.1.2.3 FUNCIONES

- Seccionar ordinariamente una vez por mes y extraordinariamente cuando el caso lo requiera.
- Proponer al H. Consejo Directivo las reformas al Reglamento Interno y normativos de este Centro.
- Planificas, coordinar y evaluar los proyectos de investigación, tesis, memorias, y prácticas de pre-grado en la Facultad, a través de comisiones de ser necesario.
- Elaborar mensualmente su plan de actividades.
- Desarrollar infraestructura investigativa.
- Realizar cursos de especialización o actualización, seminarios, mesas redondas, respecto al desarrollo de investigación y capacitación

5.1.2.4 ESTRUCTURA ORGÁNICA

La figura Figura V.1 muestra la estructura orgánica de los Centros de Investigación:

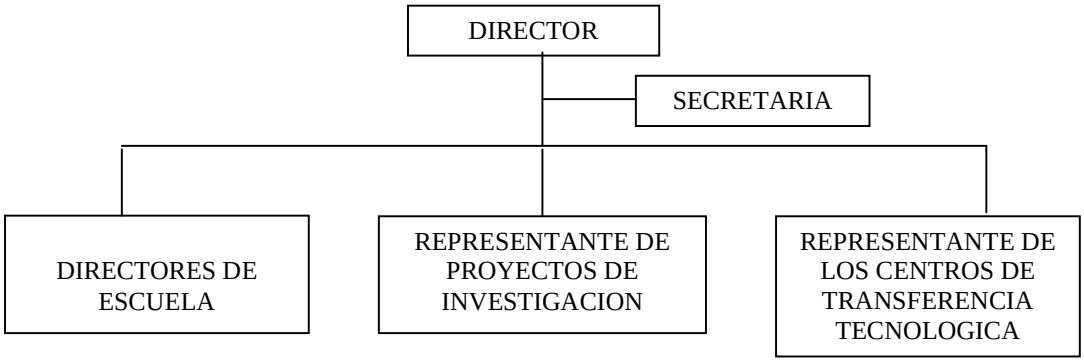


Figura V.1.- "Orgánico Funcional"

5.1.3 ANTECEDENTES TECNOLOGICOS

Siendo el Departamento de Sistemas y Telecomunicaciones (DESITEL) una dependencia de apoyo, asesoramiento y administración de los sistemas y la Red de la ESPOCH, describiremos el recurso humano, hardware y software que posee, Asi también los del CIPFIE con el fin de caracterizar los recursos que poseen estas entidades.

5.1.3.1 RECURSO HUMANO

La tabla **Tabla IV.1** muestra el recurso humano que posee el DESITEL

Tabla V.1. Recurso humano del DESITEL

Personal	Cargo
Ing. Byron Vaca	Director DESITEL
Ing. Alex Tacuri	Técnico DESITEL
Ing. Gonzalo Allauca	Técnico DESITEL

La tabla **Tabla IV.2** muestra el recurso humano que posee el CIPFIE

Tabla V.2. Recurso humano del CIPFIE

Personal	Cargo
Ing. Hugo Velastegui	Director CIPFIE
Lcda. Maria Agusta Larrea	Secretaria CIPFIE

5.1.3.2 RECURSOS HARDWARE

La tabla **IV.3** muestra las características de los recursos con los que cuenta el DESITEL:

Tabla V.3. Recurso hardware DESITEL

Recursos	Descripción
1 Servidores Intel	- 2 Discos Duros 70 GB - Procesador 3.4 Ghz - Memoria Ram 2GB - DVD R/RW
2 Servidores Intel	- 1 Discos Duros 80 GB - Procesador 2.4 Ghz - Memoria Ram 1 GB

En la tabla **IV.4.** muestra las características de los recursos hardware con los que cuenta el CIPFIE:

Tabla V.4. Recurso hardware CIPFIE

Recursos	Descripción
1 PC	- 1 Disco Duro 70 GB - Procesador 2.0 Ghz - Memoria Ram 256 MGB - CDROM
1 PC	- 1 Disco Duro 80 GB - Procesador 2.4 Ghz - Memoria Ram 256 MGB

Red Institucional:

- Una infraestructura de red Fast Ethernet (a 100Mbps) que interconecta el campo institucional y a la que están integradas casi todas sus dependencias.
- La red utiliza principalmente un cableado de cobre y se utiliza la fibra óptica para cubrir distancias grandes a manera de backbone.
- Se cuenta con equipos de conmutación que aceptan medios de cobre y fibra óptica.

Acceso satelital al Internet:

- Conexión satelital a Internet cuyo ancho de banda máximo llega a los 256kbps de subida y 512kbps de bajada.
- Acceso a Internet desde casi todos los nodos conectados a la red.

Equipos Clientes:

- Existen equipos con características heterogéneas en cuanto a hardware, a través de los cuales se brinda el acceso a Internet.

La figura Figura **V.2** muestra el esquema de la red en la institución

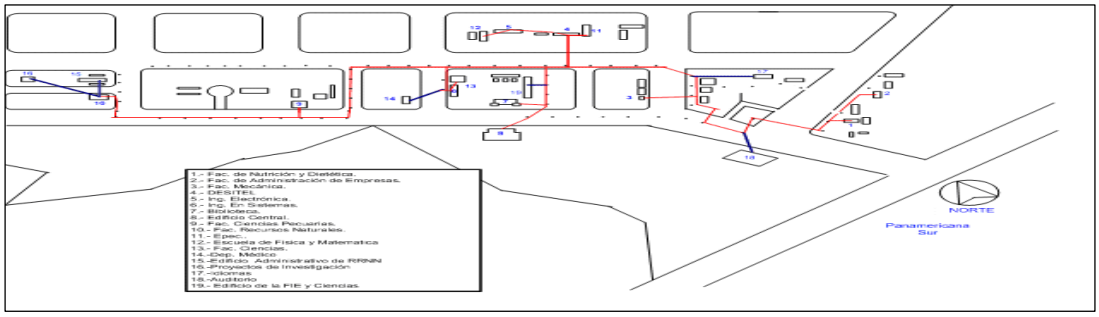


Figura V.2.- "Red Institucional"

La Figura **V.3** muestra el esquema de distribución de la red en la institución.

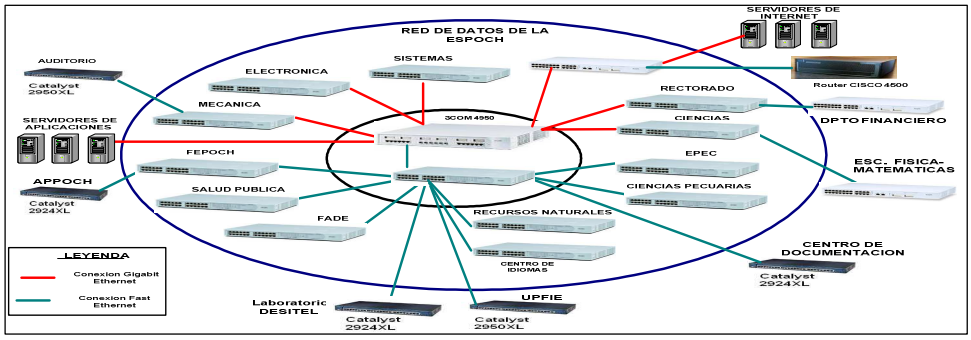


Figura V.3.- "Distribución de la red institucional"

5.1.3.3 RECURSOS SOFTWARE

La Tabla **IV.5** muestra los recursos software instalado en los servidores que posee el DESITEL mencionados anteriormente.

Tabla V.5. Recurso Software DESITEL

Recursos	Descripción	Aplicaciones
Servidor Web	- Sistema Operativo : Linux Centos 5.0 - Base datos: MySql 5.0.22 - Servidor Web: Apache 2.2.3 - PHP 5.1.6	- Sitios Web: academiaLinux.edu.espoch.ec frn.espoch.edu.ec www.espoch.edu.ec www.sangar.org
Servidor de Pruebas	- Sistema Operativo : Linux Centos 4.4 - Base datos: MySql 4.0.18 - Servidor Web: Apache 2.2.46 - PHP 4.3.2	
Servidor Proxy	- Sistema Operativo : Linux Centos 5.0 - Base datos: MySql 5.0.22 - Servidor Web: Apache 2.2.3 - PHP 5.1.6 - DHCP 3.0.5 - BIND(dns) 9.3.3 - Squid 2.6	- Becas - Cesantía - Radius(IntrenetInalambrico) - SAAE(Antares.espoch.edu.ec) - NetVisor - SARH(recursos.espoch.edu.ec)
	- Sistema Operativo : Linux Centos 5.0	

Servidor Correo	- Base datos: MySql 5.0.22 - Servidor Web: Apache 2.2.3 - PHP 5.1.6 - Postfix 2.3.2	- Sistema de Correo Electrónico de la ESPOCH
Servidor Web	- Sistema Operativo : Windows 2003 Server - Base datos: Microsoft SqlServer 2000 - Servidor Web: IIS 6.0 - Framework 1.0 - Framework 2.0	- Sistema Académico(Academico.espoch.edu.ec) - Espoch Passport passport.espoch.edu.ec

Cabe mencionar que la ESPOCH posee licenciamiento para los siguientes productos descritos en la tabla V.6:

Tabla V.6. Productos licenciados por la Espoch

Numero	Descripción del Producto
1	Windows 2003 Server Enterprise Edition
2	Microsoft Visual Studio .Net 2003
3	Microsoft SqlServer 2000

ESTUDIO DE LA FUNCIONALIDAD DEL SISTEMA ACTUAL

Para el estudio del sistema actual se efectuó visitas a las instalaciones de este organismo donde se realizó entrevistas y encuestas a los usuarios del sistema y principales funcionarios de dicho organismo. A continuación detallamos los resultados:

5.1.3.4 ENTREVISTA

Objetivo:

- Determinar la funcionalidad del sistema actual
- Identificar las entidades y usuarios del sistema actual
- Identificar las funciones de los involucrados en el sistema actual

La entrevista fue realizada a la Lcda. Maria Agusta Larrea, el día 26 de Abril del 2007. A continuación el detalle de la entrevista:

Pregunta 1:

¿Cuál es el proceso que se debe realizar para presentación y ejecución de un proyecto de tesis?

El proceso debe iniciar con la presentación del anteproyecto, su evaluación, el registro de la documentación legal del anteproyecto y su seguimiento hasta la culminación de mismo.

Pregunta 2:

¿Cuales son los requisitos y formatos para la presentación de un anteproyecto de tesis?

Los requisitos para la presentación de un anteproyecto dependen de las políticas definidas en el reglamento en vigencia del CIPFIE.

Pregunta 3:

¿Cómo se realiza la evaluación de un anteproyecto de tesis?

Luego de la presentación del anteproyecto de tesis y previa validación de requisitos, el Directorio del CIPFIE valida el tema del anteproyecto y nombra en base a los datos y

tema una comisión evaluadora, la misma que se encargara de evaluar el tema por medio de una defensa publica que lo realizara el proponente(s) del anteproyecto.
Sobre la conformación de la comisión evaluadora se detalla en el reglamento que se encuentra en vigencia del CIPFIE.

Pregunta 4:

¿Cuáles son los parámetros de evaluación para calificar un anteproyecto de tesis?

Los parámetros para la evaluación de un anteproyecto de tesis están detallados en el reglamento del CIPFIE que se encuentra en vigencia.

Pregunta 5:

¿Luego de la evaluación del anteproyecto de tesis que se debe hacer?

Si el anteproyecto fue aprobado, se debe esperar la notificación por parte del Honorable Consejo Directivo de la facultad la aprobación legal del tema y el nombramiento de un asesor para el desarrollo del proyecto en otra notificación.
Luego de la notificación del asesor designado para el desarrollo de la tesis se procederá a matricular el anteproyecto de tesis como una materia más del pensum de estudios a partir de lo cual inicia el tiempo para el desarrollo de la tesis.

Pregunta 6:

¿Cual es el tiempo máximo para el desarrollo de un proyecto de tesis?

EL tiempo máximo para el desarrollo de un proyecto de tesis depende de lo estipulado en el reglamento del CIPFIE.

Pregunta 7:

¿Qué sucede si el tiempo de desarrollo de la tesis ha terminado y no se ha culminado la investigación?

Si el tiempo de desarrollo de la tesis ha culminado el estudiante puede matricular nuevamente el anteproyecto de tesis por un periodo de 6 meses, pudiendo matricular el tema hasta un total de 3 matriculas. Si se han consumido las 3 matriculas y no se ha terminado la investigación, el estudiante puede solicitar al Honorable Consejo Directivo de la facultad una prorroga de 6 meses mas para la terminación previo la notificación de un avance del 80% por parte del tutor del proyecto.

Pregunta 8:

¿Cual es el proceso a seguir una vez terminada la investigación?

Una vez terminada la investigación el tutor del proyecto deberá notificar al CIPFIE la terminación del proyecto de investigación, y solicitará las actas respectivas para la calificación del trabajo escrito. Una vez evaluado el trabajo escrito, los resultados se enviarán al Honorable Consejo Directivo de la facultad quién analizará la documentación y emitirá una notificación donde fijara fecha de defensa oral de la investigación.

Las actas para registrar las calificaciones de la defensa oral y del trabajo escrito el tutor las solicitará en la secretaria del CIPFIE. Una vez registradas las notas del trabajo escrito y la defensa oral, la secretaria del CIPFIE registrará la culminación de la investigación.

Pregunta 9:

¿Qué métodos y herramientas utilizan para la realización de cada uno de los procesos?

Algunas actividades se las realiza de forma semiautomática con la ayuda de hojas de cálculo de Excel para registrar los proyectos y tiempos. Para la realización de oficios y notificaciones se utiliza Word y para el registro y seguimiento de proyectos se los realiza de forma manual, convirtiéndose esta en una actividad muy tediosa y con la perdida de mucho tiempo.

Pregunta 10:

¿Existen actividades complementarias a la gestión de los proyectos de tesis?

Si la actividad complementaria a la gestión de los proyectos de tesis es la de gestionar la información promocional de los proyectos de tesis, la que consiste en la clasificación de los proyectos de tesis por el área a la que pertenecen para una promoción adecuada.

5.1.3.5 DESCRIPCIÓN DEL SISTEMA ACTUAL

En base a los datos obtenidos en las actividades anteriores de la metodología podemos describir la funcionalidad del sistema actual.

Los actores involucrados en las diferentes actividades de los procesos son:

- ***Estudiantes***, quienes son los proponentes de los proyectos de tesis.
- ***Profesores***, encargados de guiar la investigación de propuesta por un estudiante en el caso de ser elegido *Director de tesis o Asesor.*

- ***Comisión Evaluadora***, encargada de evaluar los anteproyectos de tesis propuestos por los estudiantes.
- ***Secretaria CIPFIE***, encargada de registrar toda la información de proyectos y anteproyectos de tesis, así también proporcionar información de la misma.
- ***Director CIPFIE***, encargado de vigilar la correcta funcionalidad de todos los procesos desarrollados por el CIPFIE, así también de analizar los anteproyectos presentados y de nombrar la comisión evaluadora para calificar los anteproyectos.
- ***Honorable Consejo Directivo*** de la facultad, quien es encargado de proporcionar el aval legal de ciertas actividades como aprobación del anteproyecto de tesis, nombrar tribunal asesor y otras.
- ***Publico en General***, quien solicita información sobre proyectos de investigación realizados y otros.

Los procesos identificados son:

- ***Gestión de anteproyectos de tesis:*** Las actividades que se realizan en este proceso son las siguientes:
    1. El estudiante entrega los documentos del anteproyecto al centro de Investigación.
    2. La secretaria del centro de investigación analiza la información
    3. El Director del CIPFIE revisa el tema, asigna y convoca a la comisión técnica para la evaluación del anteproyecto.
    4. El estudiante expone el tema ante la comisión técnica designada.
    5. La comisión técnica evalúa el anteproyecto.
    6. La comisión técnica envía los resultados de la evaluación al Director del CIPFIE.

7. El centro de investigación informa al Honorable Consejo Directivo Directivo de la facultad los resultados.
8. El Honorable Consejo Directivo analiza y designa asesor para el proyecto de tesis, así también emite orden de matricula para el proyecto de tesis.
9. El estudiante matricula el proyecto de tesis e inicia el tiempo para el desarrollo de la tesis.

- ***Gestión de proyectos de tesis:***

1. El Director de Tesis notifica a la secretaria del CIPFIE la culminación de la investigación.
2. La secretaria emite actas para evaluación del trabajo escrito.
3. El tribunal asesor evalúa el trabajo escrito y envía al CIPFIE.
4. El Director del CIPFIE, envía al Honorable Consejo Directivo para su análisis respectivo, el Honorable Consejo Directivo designa fecha de defensa oral del proyecto.
5. La secretaria del CIPFIE, previa notificación de la fecha de defensa oral del proyecto de tesis emite las actas para el registro de la evaluación oral del proyecto de tesis. Una vez emitido las actas para la defensa oral, la secretaria registra la culminación de la investigación.

- ***Gestión Anular Proyecto de tesis:***

1. El Director de Tesis solicita anulación del proyecto de tesis al director del CIPFIE.
2. El Director del CIPFIE notifica al C.D la anulación del proyecto de tesis
3. EL C.D analiza solicitud y anulación proyecto de tesis
4. EL C.D notifica anulación de proyecto de tesis

- ***Gestión Cambio Asesor proyecto de tesis:***

    1. El estudiante solicita cambio de miembro tribunal asesor al Director del CIPFIE.
    2. El Director del CIPFIE notifica al C.D el cambio de tribunal asesor.
    3. El C.D analiza solicitud de cambio de miembro tribunal asesor.
    4. El C.D notifica cambio de miembro tribunal asesor de proyectos de tesis.

- ***Gestión Información Disponibilidad Director de Tesis:***

    1. El estudiante solicita información sobre cupo Director Tesis a secretaria del CIPFIE.
    2. La secretaria del CIPFIE analiza solicitud.
    3. La secretaria del CIPFIE presenta información requerida.

- ***Gestión Solicitud de Prorroga para proyecto de tesis:***

    1. El Director de tesis solicita prorroga para proyecto de tesis al Director del CIPFIE.
    2. El Director de tesis notifica al C.D solicitud de prorroga para proyecto de tesis.
    3. El C.D analiza solicitud de prorroga para proyecto de tesis.
    4. EL C.D notifica prorroga para proyecto de tesis.

- ***Gestión Solicitud de Matricula para proyecto de tesis:***

    1. El estudiante solicita matricula de proyecto de tesis al Director del CIPFIE
    2. El Director del CIPFIE, notifica al C.D solicitud de matricula para proyecto de tesis
    3. El C.D analiza solicitud de matricula para anteproyecto de tesis

4. EL C.D notifica y emite orden de matricula para proyecto de tesis

- ***Gestión de la información de los proyectos de tesis:***

1. El estudiante o publico en general acude al centro de investigación y solicita información
2. La secretaria del CIPFIE analiza la petición y busca la información
3. La secretaria del CIPFIE informa los resultados de la búsqueda
4. La secretaria del CIPFIE entrega información solicitada al cliente
5. El estudiante o público en general recibe la información solicitada del CIPFIE.

IDENTIFICACION DE ESENARIOS PROBLEMA

Los principales problemas percibidos los podemos detallar en la tabla V.7:

Tabla V.7. Problemas identificado en los procesos identificados.

PROBLEMAS	PROCESO AFECTADO
Falta de información acerca de las funciones que provee el CIPFIE.	Gestión anteproyectos de tesis Gestión proyectos de tesis Gestión información de proyectos de tesis
Falta de métodos actuales para el manejo de la información.	Gestión anteproyectos de tesis Gestión proyectos de tesis Gestión información de proyectos de tesis
Falta de promoción de proyectos realizados y en realización.	Gestión información de proyectos de tesis
Temas repetidos, información duplicada, redundancia de datos.	Gestión anteproyectos de tesis

Información almacenada en archivos físicos, sujeta a perdidas.	Gestión anteproyectos de tesis Gestión proyectos de tesis Gestión información de proyectos de tesis
Falta de seguimiento de las investigaciones por parte de las autoridades.	Gestión anteproyectos de tesis Gestión proyectos de tesis Gestión información de proyectos de tesis
Procedimientos tediosos y manuales, lo cual influye en el tiempo de productividad.	Gestión anteproyectos de tesis Gestión proyectos de tesis Gestión información de proyectos de tesis
Falta de información sobre los tramites a seguir para la presentación de los anteproyectos.	Gestión anteproyectos de tesis Gestión proyectos de tesis
Redundancia de datos.	Gestión anteproyectos de tesis
Inseguridad en la información.	Gestión anteproyectos de tesis Gestión proyectos de tesis
Excesivos tiempos en búsqueda de la información.	Gestión anteproyectos de tesis Gestión proyectos de tesis
Falta de mecanismos de difusión.	Gestión proyectos de tesis

5.2 FASE 2: MODELAMIENTO DEL NEGOCIO

5.2.1 PLANIFICACIÓN PRELIMINAR DE LA ETAPA

5.2.1.1 OBJETIVOS DEL MODELAMIENTO DEL NEGOCIO

Identificar y modelar los procesos de negocio para la gestión y control de los anteproyectos y proyectos de tesis del CIPFIE, permitiendo identificar problemas para su resolución por medio de la automatización de estos procesos por medio de una tecnología de servicios.

5.2.1.2 FORMAR EL EQUIPO DE TRABAJO

DIRECTOR:

Ing. Gloria Arcos

MIEMBROS:

Ing. Danilo Pastor

Ing. Iván Menes

Sr. Juan Pablo Tixe

Sr. Luis Rivera

USUARIOS:

Lcda. Maria Agusta Larrea

5.2.2 DEFINIR LOS LIMITES DEL PROCESO

5.2.2.1 DESCRIPCION DE LOS PROCESOS

Según la información levantada en las etapas anteriores de la metodología describimos los procesos identificados.

PROCESO: GESTIÓN DE ANTEPROYECTOS DE TESIS

Tabla V.8. Gestión de anteproyectos de Tesis

Caso de Uso	CU-PCIPFIE-1
Nombre	Gestionar anteproyectos de tesis.
Actores	Estudiante, Secretaria CIPFIE, Director CIPFIE, Comisión Evaluadora, Tribunal Asesor, C.D.
Función	Gestionar etapas de un anteproyecto de tesis.
Descripción	El proceso inicia cuando el estudiante presenta su anteproyecto de tesis en la secretaria del CIPFIE, donde luego de ser revisados los requisitos y las restricciones del caso (tema no repetido, requisitos académicos del

	estudiante, certificado de la empresa), el anteproyecto es analizado por el directorio del CIPFIE, quien asigna una comisión evaluadora que calificará el anteproyecto, si el anteproyecto es aprobado(según parámetros de evaluación definidos en el reglamento vigente del CIPFIE) el anteproyecto será aprobado legalmente por parte del C.D que en una primera instancia designará un miembro para el tribunal asesor del proyecto de tesis y en una segunda instancia autorizará orden de matricula para el proyecto de tesis.
Caso Alternativo	Si en la evaluación el anteproyecto de tesis no es aprobado se procederá a su anulación.

PROCESO: GESTIÓN DE PROYECTOS DE TESIS

Tabla V.9. Gestión de proyectos de tesis

Caso de Uso	CU-PCIPFIE-2
Nombre	Gestionar proyecto de tesis.
Actores	Director Tesis, Director CIPFIE, Tribunal Asesor, C.D, Secretaria CIPFIE.
Función	Gestionar cambio de asesor de un proyecto de tesis.
Descripción	El proceso inicia cuando el Director de Tesis notifica al Director del CIPFIE la culminación de la investigación, la secretaria del CIPFIE emite actas para evaluación del trabajo escrito, el tribunal asesor evalúa el trabajo escrito y envía al CIPFIE, el Director del CIPFIE envía al Honorable Consejo Directivo para su análisis respectivo, el Honorable Consejo Directivo designa delegado para evaluación y fecha de defensa oral del proyecto, la secretaria del CIPFIE, previa notificación de la fecha de defensa oral del proyecto de tesis emite las actas para el registro de la

	evaluación oral del proyecto de tesis. Una vez emitido las actas para la defensa oral, la secretaria del CIPFIE registra la culminación de la investigación.

PROCESO: GESTIÓN INFORMACIÓN DE PROYECTOS DE TESIS

Tabla V.10. Gestión información de proyectos de tesis

Caso de Uso	CU-PCIPFIE-3
Nombre	Gestionar información de los proyectos de Tesis.
Actores	Estudiante, Secretaria CIPFIE.
Función	Solicitar Información de proyectos de tesis del CIPFIE.
Descripción	El proceso inicia cuando el estudiante solicita información de los proyectos de tesis a la secretaria del CIPFIE, la secretaria del CIPFIE analiza la solicitud presentada por el estudiante, la secretaria del CIPFIE presenta la información solicitada.

PROCESO: GESTIÓN INFORMACIÓN DISPONIBILIDAD DIRECTOR DE TESIS

Tabla V.11. Gestión información disponibilidad director de tesis

Caso de Uso	CU-PCIPFIE-4
Nombre	Gestionar información disponibilidad director de tesis.
Actores	Estudiante, Secretaria CIPFIE.
Función	Solicitar Información sobre disponibilidad de profesor para director de tesis.
Descripción	El proceso inicia cuando el estudiante solicita información sobre disponibilidad de algún profesor para director de tesis a la secretaria del CIPFIE, la secretaria del CIPFIE analiza la solicitud presentada por el estudiante, la secretaria del CIPFIE presenta la información solicitada.

PROCESO: GESTIÓN CAMBIO ASESOR PROYECTO DE TESIS

Tabla V.12. Gestión cambio Asesor proyecto de tesis

Caso de Uso	CU-PCIPFIE-5
Nombre	Gestionar cambio asesor proyecto tesis.
Actores	Estudiante, Decano de la facultad, C.D.
Función	Gestionar cambio de asesor de un proyecto de tesis.
Descripción	El proceso inicia cuando el estudiante solicita al Decano de la facultad cambio de miembro del tribunal asesor del proyecto de tesis, el Decano de la facultad notifica dicha solicitud al C.D de la facultad para su análisis, el C.D de la facultad analiza la solicitud presentada por el estudiante, el C.D emite notificación de resultado de la solicitud del estudiante.
Condiciones	El proyecto de tesis debe estar aprobado y se debe haber designado tribunal asesor.

PROCESO: GESTIÓN ANULAR PROYECTO DE TESIS

Tabla V.13. Gestión anular proyecto de tesis

Caso de Uso	CU-PCIPFIE-6
Nombre	Gestionar anulación proyecto de tesis.
Actores	Director Tesis, Director CIPFIE, C.D, Secretaria del CIPFIE.
Función	Gestiona la anulación de un proyecto de tesis.
Descripción	El proceso inicia cuando el Director de Tesis solicita al Director del CIPFIE la anulación de un proyecto de tesis, el Director del CIPFIE notifica la solicitud de anulación de proyecto de tesis al C.D, el C.D analiza solicitud de anulación de proyecto de tesis, el C.D notifica la anulación de proyecto de tesis, la Secretaria del CIPFIE registra la anulación del proyecto de tesis.

PROCESO 7: GESTIÓN SOLICITUD DE MATRICULA DE PROYECTO DE TESIS

Tabla V.14. Gestión solicitud de matrícula de proyecto de tesis

Caso de Uso	CU-PCIPFIE-7
Nombre	Gestionar matricula de proyecto de tesis.
Actores	Estudiante, Director CIPFIE, C.D.
Función	Gestiona la matricula de un proyecto de tesis.
Descripción	El proceso inicia cuando el Estudiante solicita al Director del CIPFIE matricula para proyecto de tesis, el Director del CIPFIE notifica al C.D la solicitud de matricula por parte del Estudiante, el C.D analiza solicitud de matricula, el C.D notifica la autorización de matricula para proyecto de tesis al Estudiante, El estudiante compra matricula, el Estudiante notifica la matricula del proyecto de tesis al Director del CIPFIE, la secretaria del CIPFIE registra matricula del proyecto de tesis.
Condiciones	Si no es la primera matricula, el estudiante deberá poseer el número de matricula permitido para realizar matricula.

PROCESO: GESTIÓN SOLICITUD DE PRORROGA PARA PROYECTO DE TESIS

Tabla V.15. Solicitud de prórroga para proyecto de tesis

Caso de Uso	CU-PCIPFIE-8
Nombre	Gestionar prorroga para proyecto de tesis.
Actores	Director De Tesis, Director CIPFIE, C.D.
Función	Gestiona la solicitud de prorroga para un proyecto de tesis.
Descripción	El proceso inicia cuando el Director de Tesis solicita una prorroga para proyecto de tesis al Director del CIPFIE, el Director del CIPFIE notifica al C.D solicitud de prorroga para proyecto de tesis, el C.D analiza solicitud, el C.D notifica prorroga para proyecto de tesis al Director de Tesis del proyecto solicitado.
Condiciones	El estudiante deberá poseer el número de matricula permitido y tener un porcentaje de avance de un mínimo de 80%.

5.2.2.2 DEFINIR ÁMBITO DE ACCIÓN

La tabla V.16 presenta la estructura jerárquica de los procesos permitiendo definir el ámbito de acción, de esta manera identificaremos los procesos de negocio claves de nuestro dominio de problema.

Tabla V.16. Estructura Jerárquica del proceso

Estructura Jerárquica Del proceso	1. Proceso
	2. Subproceso 3. Actividades
	4. Tareas

Una vez analizado la estructura funcional y la información obtenida en las fases anteriores de la metodología, se identificó los siguientes procesos motivo de nuestro análisis.

La figura V.4.- muestra los procesos del CIPFIE para nuestro dominio de estudio.

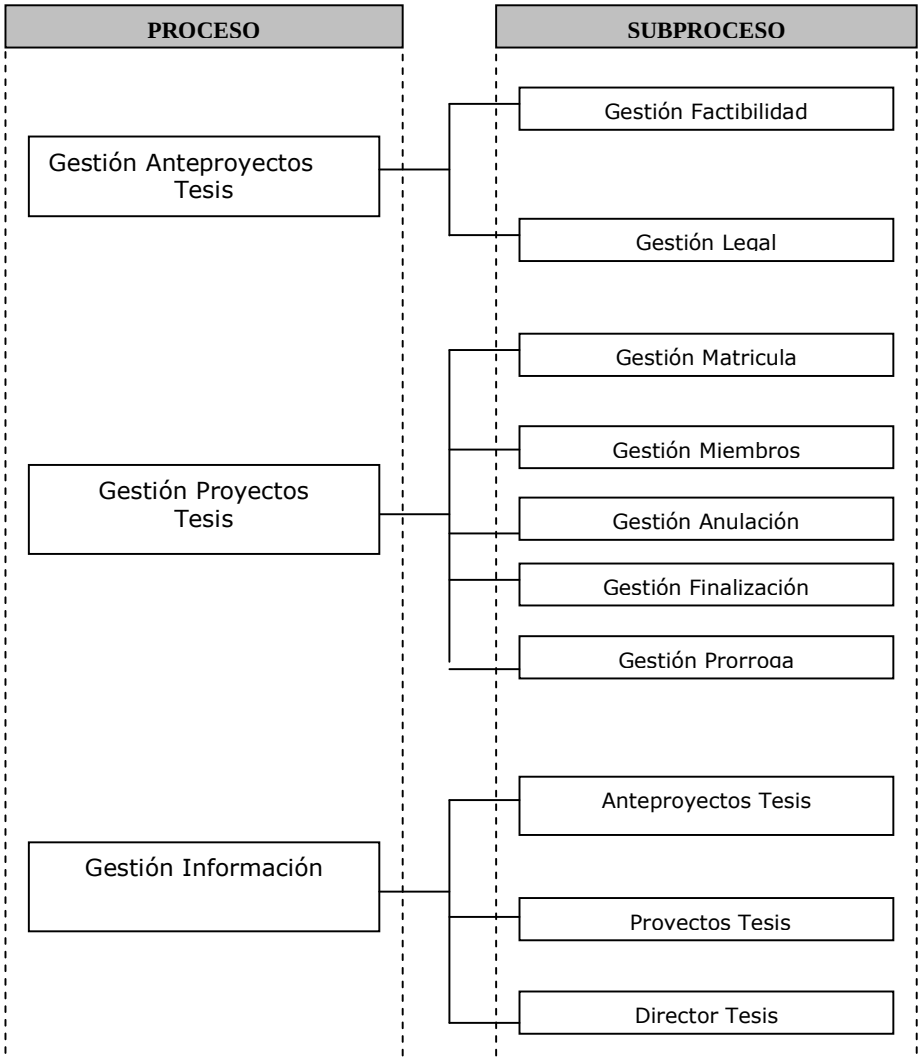


Figura V.4.- "Principales procesos del CIPFIE"

CATALOGO DE PROCESOS

En base a la información de los procesos levantados en las fases anteriores de la metodología procedemos a levantar un catálogo o inventario de procesos del CIPFIE contemplados en nuestro dominio de problema.

Tabla V.17. Catálogo de Procesos

CATALOGO DE PROCESOS		
PROCESOS	**SUBPROCESOS**	***ACTIVIDADES***
GESTIÓN ANTEPROYECTO DE TESIS	GESTION FACTIBILIDAD	Revisión y verificación de documentos para anteproyectos.
		Designación de la comisión evaluadora.
		Emisión de convocatorias para evaluación.
		Realización de la evaluación.
		Firmas de convocatorias.
		Revisión y verificación de informe de evaluación.
		Archivar Documentación.
GESTION PROYECTO DE TESIS	GESTION MATRICULA	Designación de miembro asesor de proyecto de tesis.
		Notificación de miembro de tribunal asesor.
		Matriculación de proyecto de tesis.
	GESTION CAMBIO MIEMBROS	Presentación de solicitud.
		Revisión y análisis de solicitud
		Notificación cambio de miembro de proyecto de tesis.
		Archivar resolución final.

Tabla V.17. (Continuación)

CATALOGO DE PROCESOS		
PROCESOS	**SUBPROCESOS**	**ACTIVIDADES**
	GESTION ANULACION	Presentación de solicitud.
		Revisión y análisis de solicitud.
		Notificación de anulación de proyecto de tesis.
		Archivar resolución final.
		Elaboración del informe de tesis

GESTION PROYECTO DE TESIS	GESTION FINALIZACION	culminada.
		Emisión de actas para evaluación escrita de tesis.
		Revisión y verificación de actas calificadas.
		Revisión y verificación del documento terminado de la tesis.
		Verificación de requisitos previos a la defensa publica.
		Emisión de informe de evaluación de actas escrita de tesis.
		Emisión de actas para evaluación oral final.
	GESTION PRORROGA	Presentación de solicitud para prorroga de proyecto de tesis.
		Revisión y análisis de solicitud.
		Notificación prorroga aprobada/negada.
		Archivar resolución final.
GESTION INFORMACION	ANTEPROYECTOS TESIS	Solicitud información de anteproyectos de tesis.
		Análisis de solicitud.
		Emisión de resultados de solicitud.
	PROYECTOS DE TESIS	Solicitud información de proyectos de tesis.
		Emisión de resultados de solicitud.
	DIRECTORES DE TESIS	Solicitud información de profesores directores de tesis.
		Análisis de solicitud.
		Emisión de resultados de solicitud.

5.2.2.3 CATALOGO DE ACTORES

La Tabla V.18 presenta los actores involucrados en las diferentes actividades de los procesos mencionados anteriormente:

Tabla V.18. Catálogo de Actores

Actores	Descripción
Estudiante	Son quienes proponen y realizan los proyectos de tesis.
Director Tesis	Profesor de la facultad encargado de guiar un proyecto de tesis.
Comisión Evaluadora	Profesores de la facultad designados por el Director del CIPFIE para la evaluación de un anteproyecto de tesis.
Secretaria CIPFIE	Empleada del CIPFIE encargada de llevar y registrar la información de los anteproyectos y proyectos de tesis.
Director CIPFIE	Encargado de vigilar la correcta funcionalidad de todos los procesos desarrollados por el CIPFIE, así también de analizar los anteproyectos presentados y de nombrar la comisión evaluadora para calificar los anteproyectos.
Honorable Consejo Directivo (C.D)	Quien es encargado de proporcionar el aval legal de ciertas actividades como aprobación del anteproyecto de tesis, nombrar tribunal asesor y otras.
Tribunal Asesor	Profesor de la facultad delegado por el C.D para el asesoramiento de un proyecto de tesis.

5.2.2.3.1 DESCUBRIR EL PROCESO

5.2.2.3.1.1 MODELAR EL PROCESO

5.2.2.3.1.2 DIAGRAMAS DE FLUJO DE DATOS

PROCESO: GESTION ANTEPROYECTO DE TESIS – DIAGRAMAS INDIVIDUALES

PROCESO: GESTIÓN ANTEPROYECTO DE TESIS **SUBPROCESO:** GESTION FACTIBILIDAD	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES: Estudiante, Secretaria CIPFIE, Comisión Evaluadora, Directorio CIPFIE	

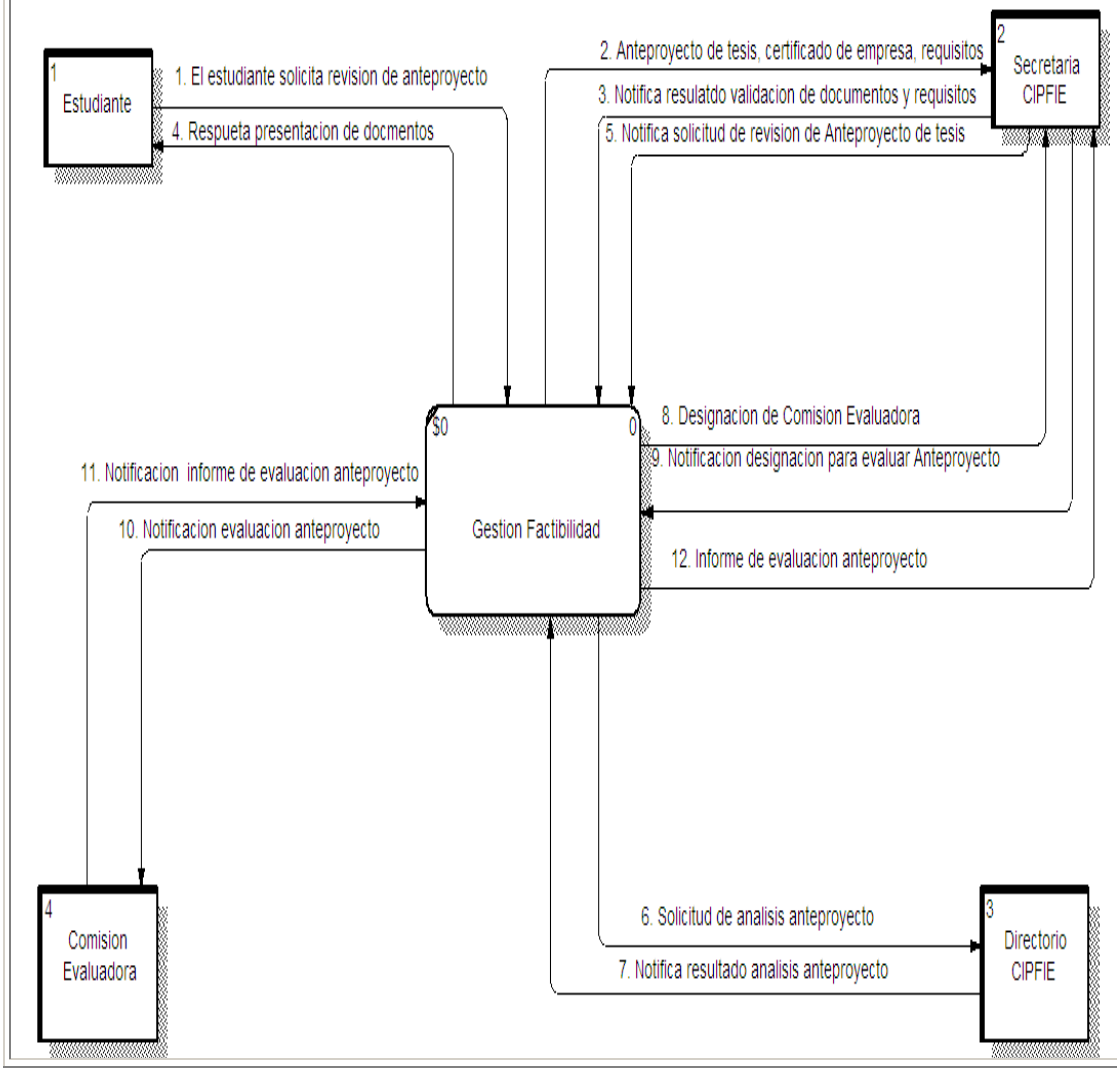


Figura V.5.- "Subproceso: Gestión factibilidad"

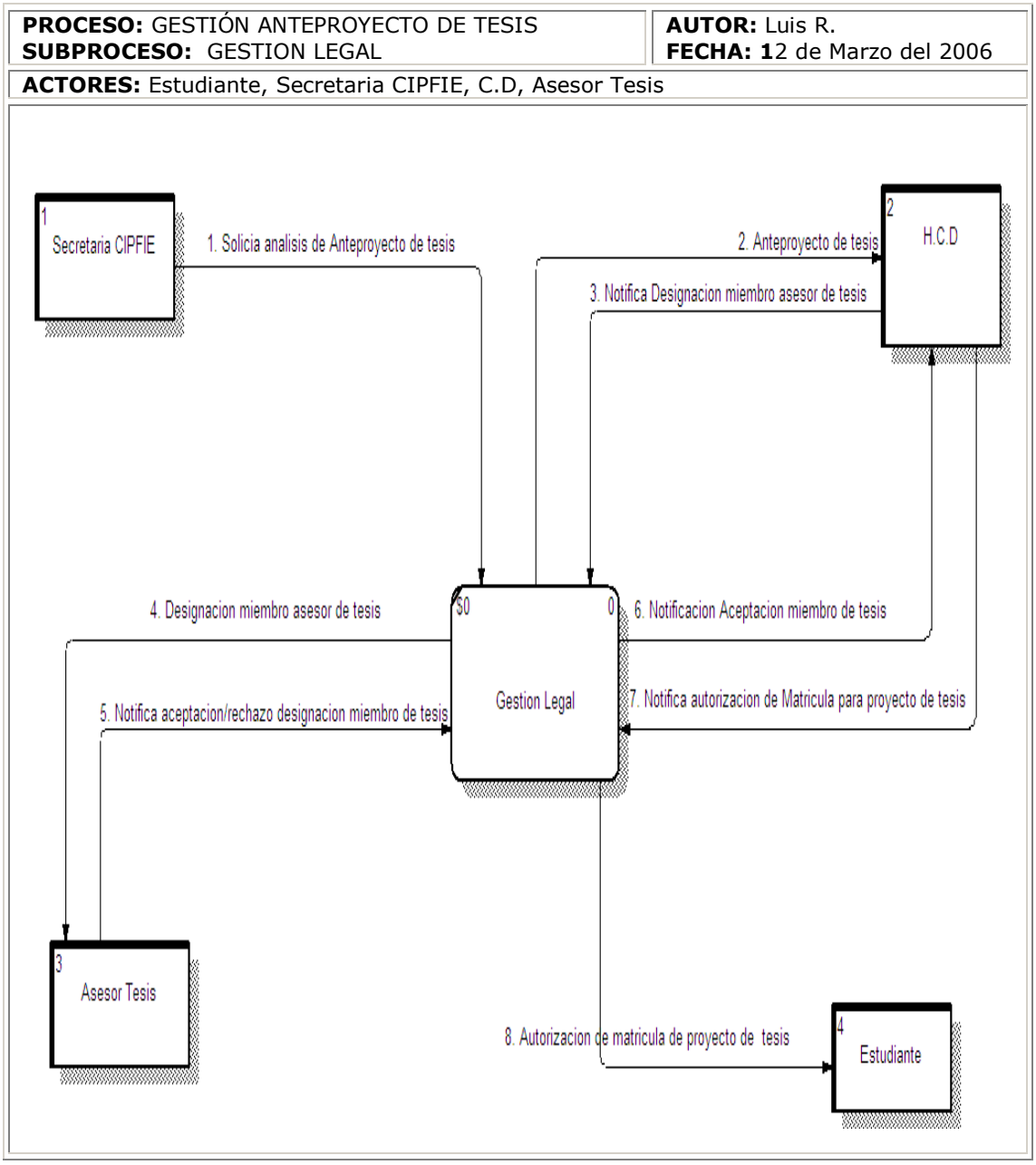


Figura V.6.- "Subproceso: Gestión Legal"

PROCESO: GESTION PROYECTO DE TESIS – DIAGRAMAS INDIVIDUALES

PROCESO: GESTIÓN PROYECTO DE TESIS **SUBPROCESO:** GESTION MATRICULA	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES: Estudiante, Secretaria CIPFIE, C.D	

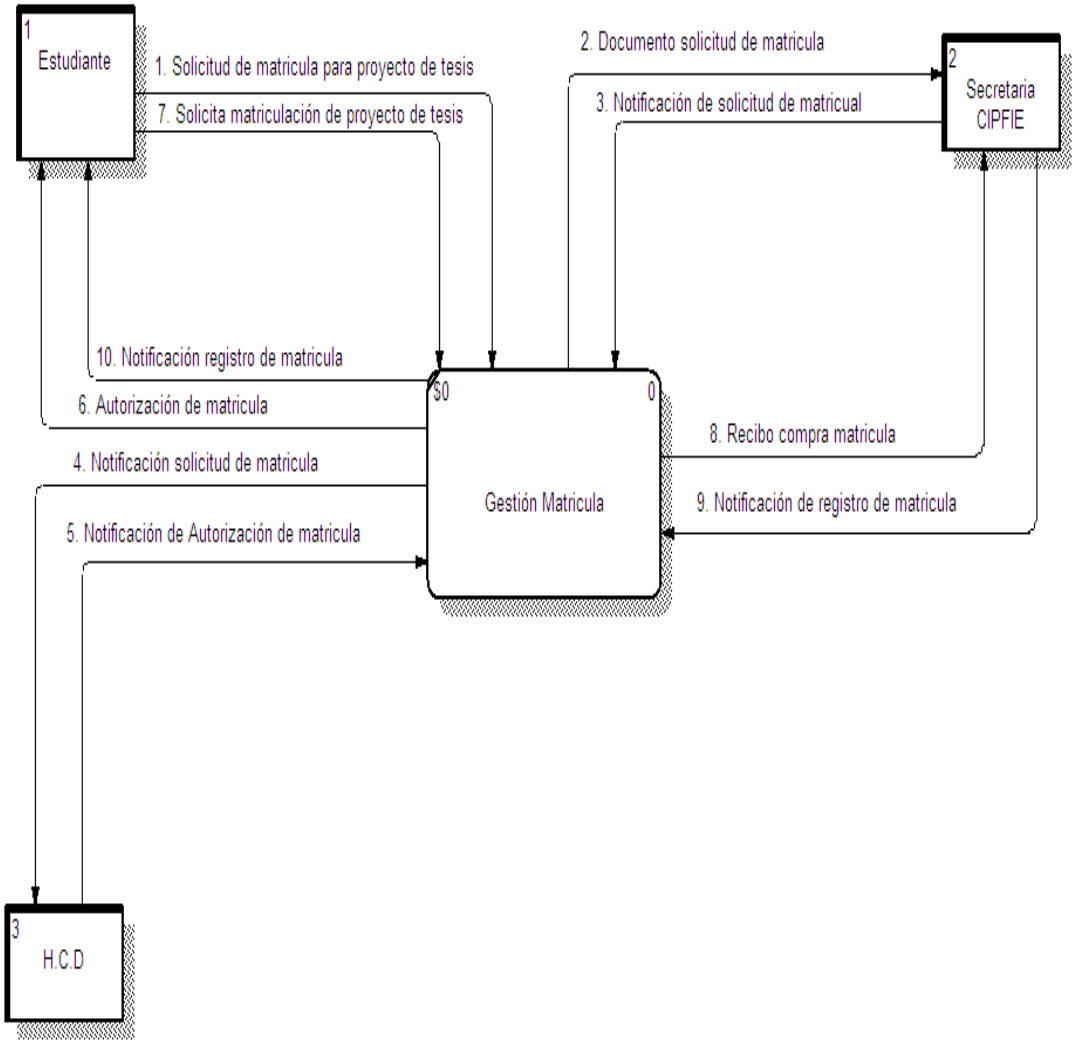


Figura V.7.- "Subproceso: Gestión Matricula"

PROCESO: GESTIÓN PROYECTO DE TESIS **SUBPROCESO:** GESTION MIEMBROS	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES: Estudiante, Secretaria CIPFIE, C.D	

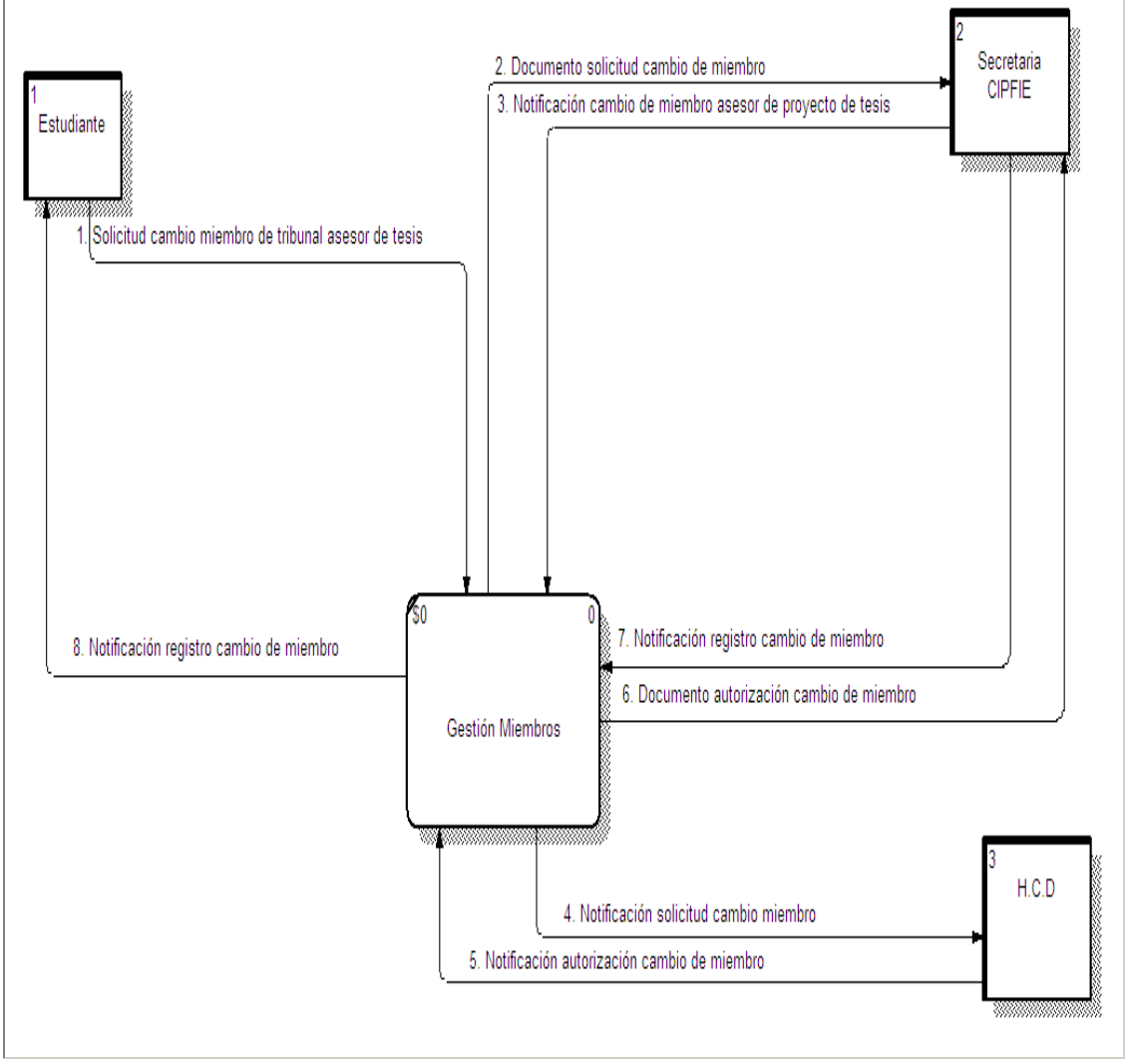


Figura V.8.- "Subproceso: Gestión Miembros"

PROCESO: GESTIÓN PROYECTO DE TESIS **SUBPROCESO:** GESTION ANULACION	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES: Director Tesis, Secretaria CIPFIE, Directorio CIPFIE	

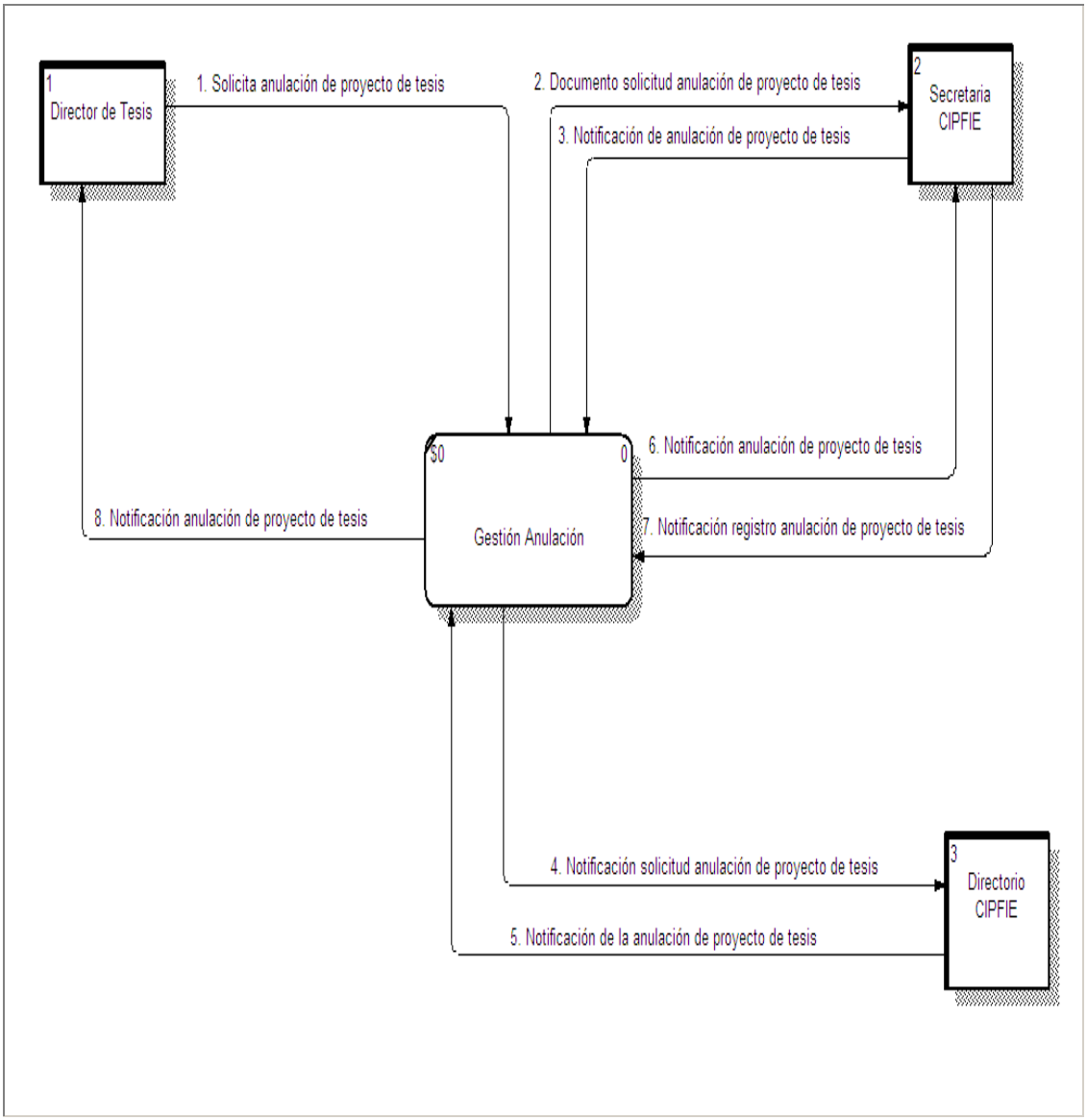


Figura V.9.- "**Subproceso:** Gestión Anulación"

PROCESO: GESTIÓN PROYECTO DE TESIS **SUBPROCESO:** GESTION FINALIZACION	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES: Estudiante, Director Tesis, Secretaria CIPFIE, Tribunal Asesor, C.D, Secretaria Escuela, Delegado Decano de la Facultad.	

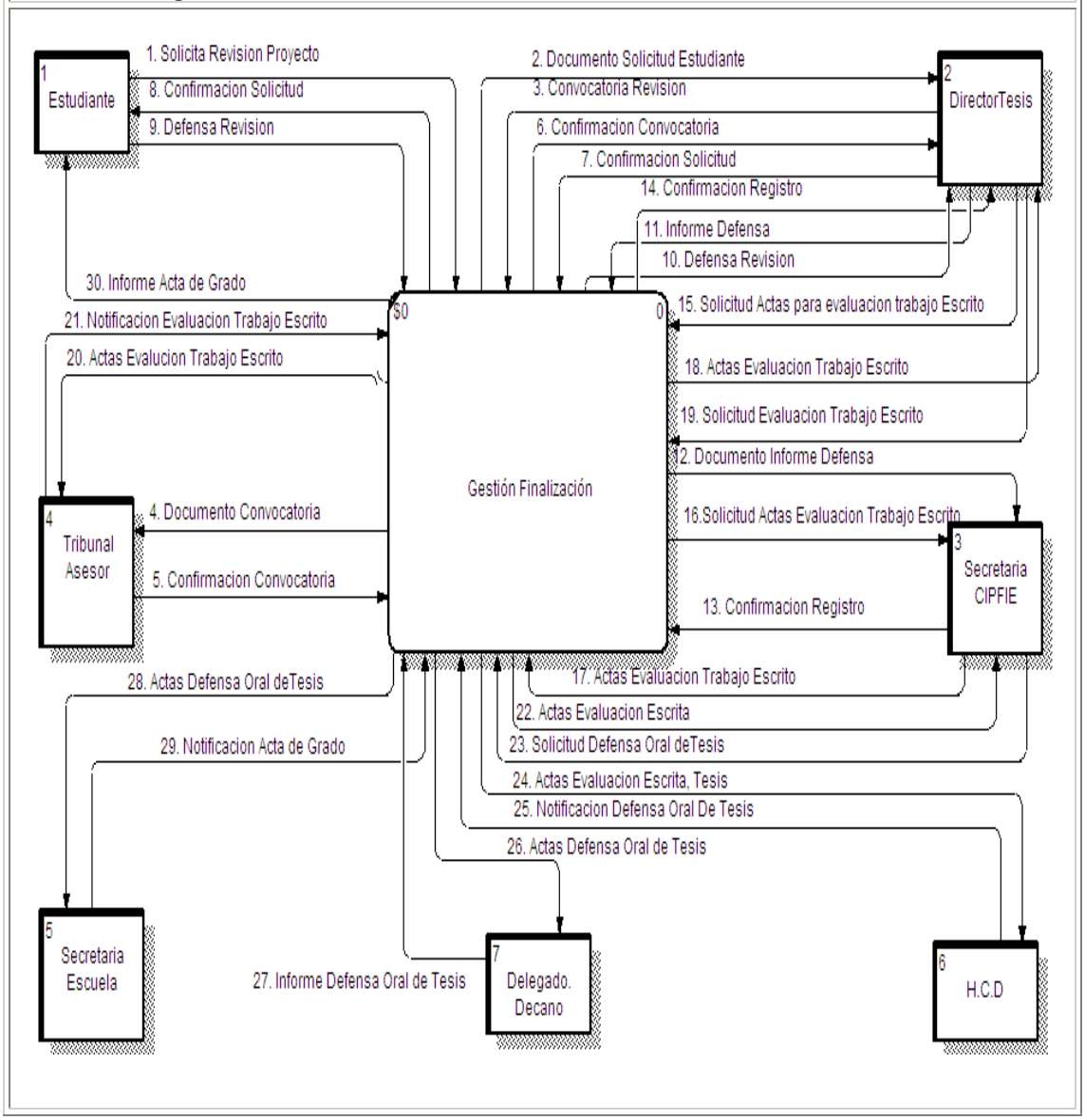


Figura V.10.- "Subproceso: Gestión Finalización"

PROCESO: : GESTIÓN PROYECTO DE TESIS **SUBPROCESO:** GESTION PRORROGA	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES Director de tesis, Secretaria CIPFIE, C.D.	

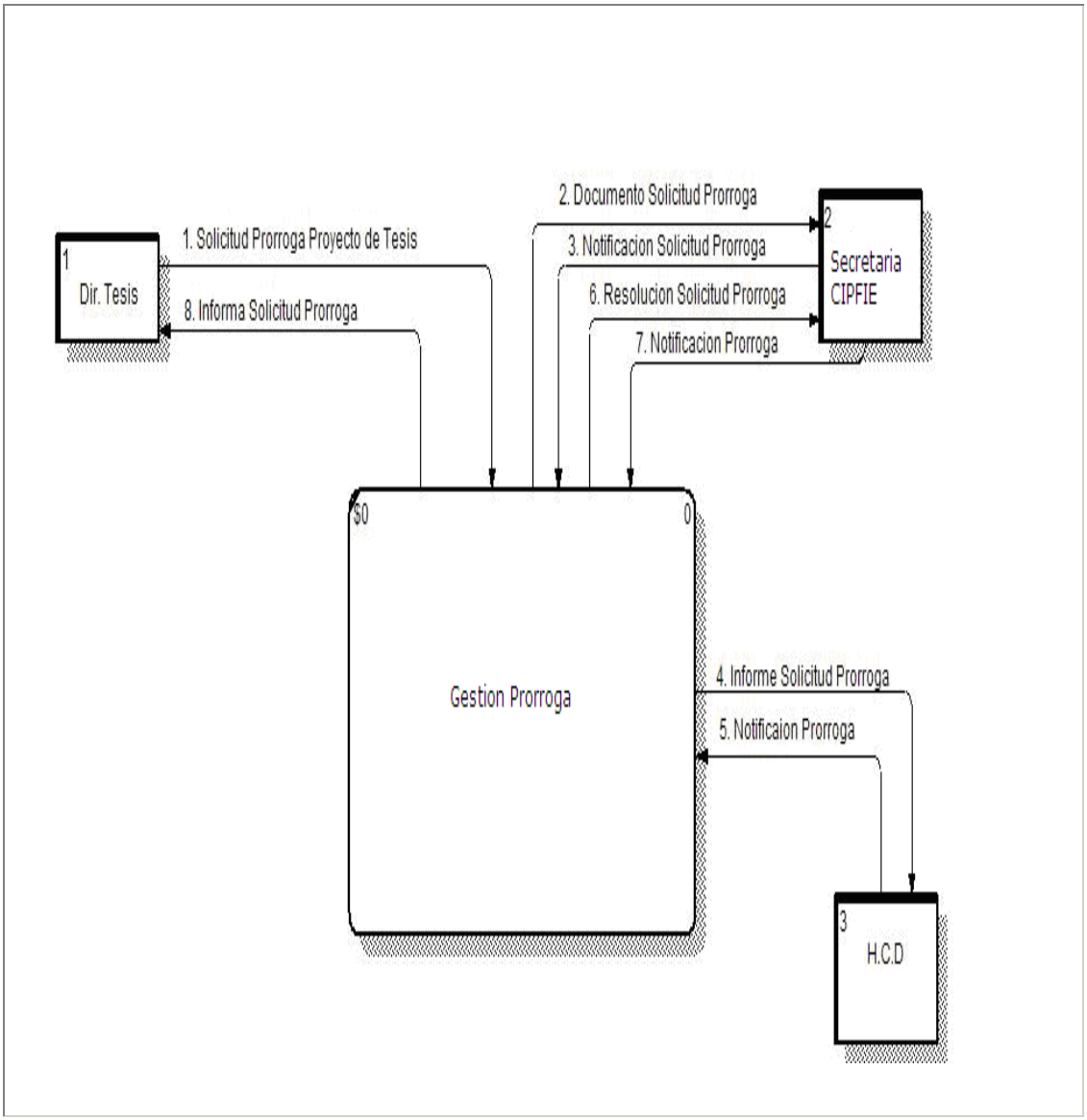


Figura V.11.- "Subproceso: Gestión Prorroga"

PROCESO: GESTION INFORMACION – DIAGRAMAS INDIVIDUALES

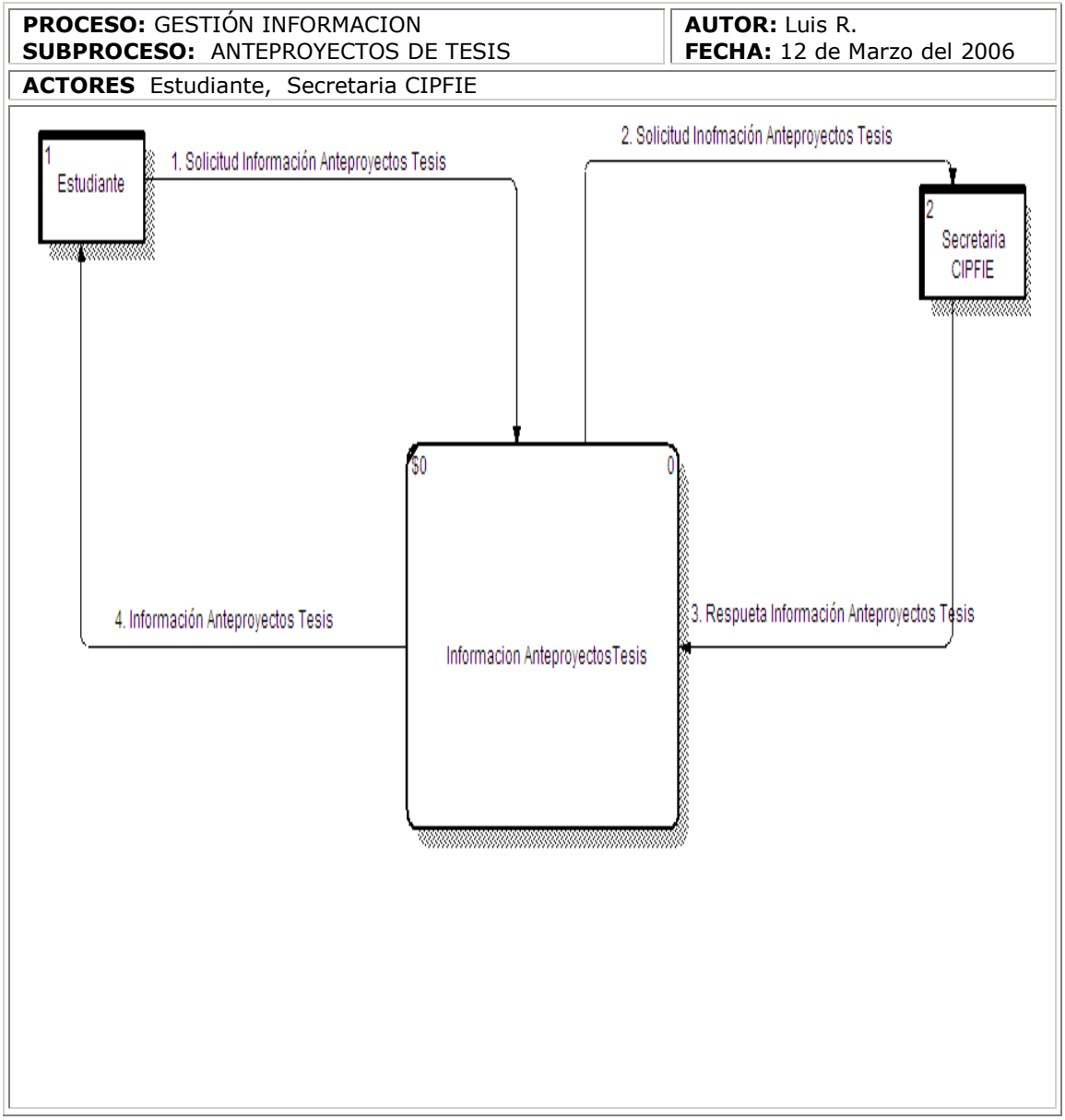


Figura V.12.- "**Subproceso:** Anteproyectos de tesis"

PROCESO: GESTIÓN INFORMACION **SUBPROCESO:** PROYECTOS DE TESIS	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE	

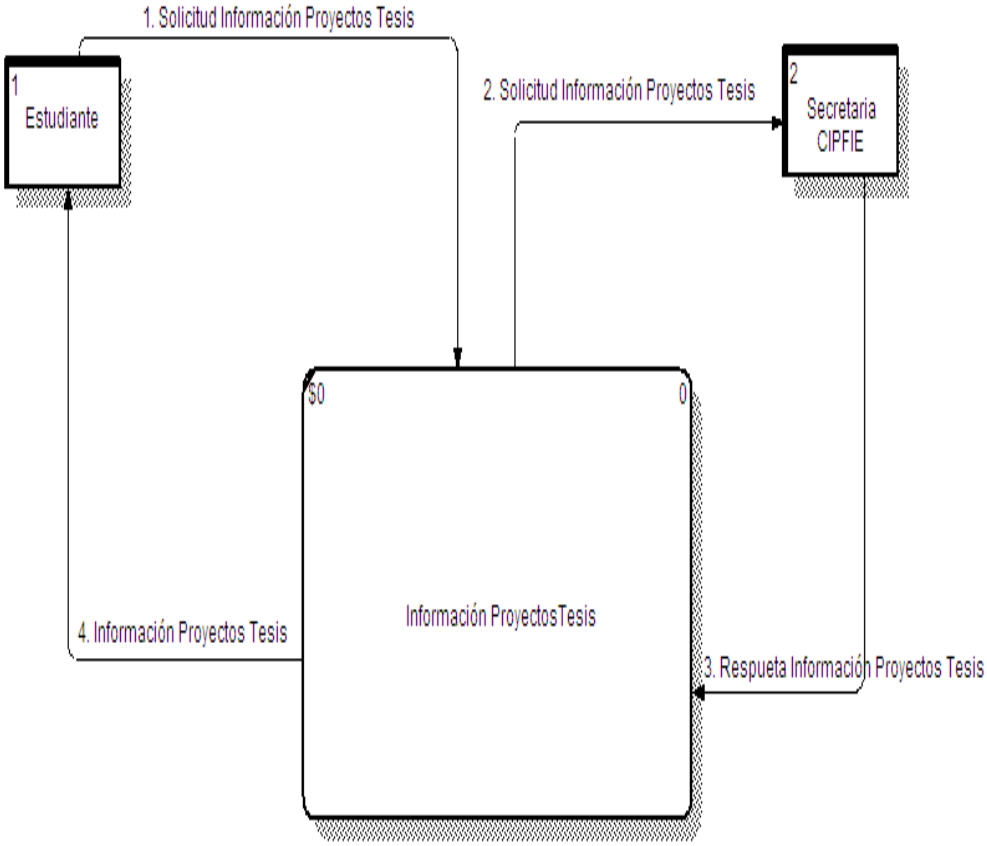


Figura V.13.- "Subproceso: Proyectos de tesis"

PROCESO: GESTIÓN INFORMACION **SUBPROCESO:** DIRECTORES DE TESIS	**AUTOR:** Luis R. **FECHA:** 12 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE	

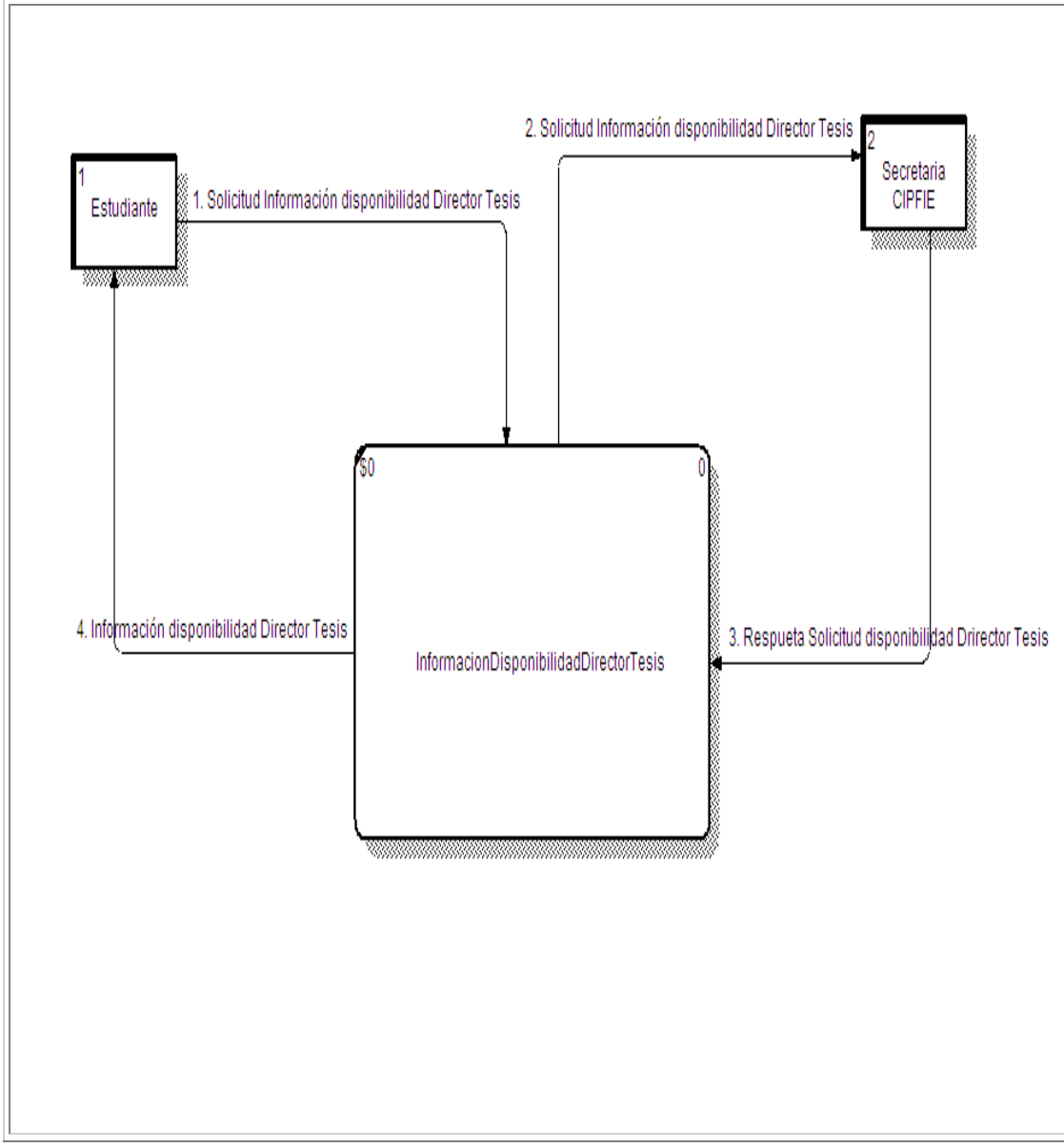


Figura V.14.- "Subproceso: Directores de tesis"

PROCESO: GESTION ANTEPROYECTO DE TESIS - DIAGRAMA INTEGRADO

PROCESO: GESTIÓN ANTEPROYECTO DE TESIS	**AUTOR:** Luis R. **FECHA:** 12 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE, Comisión Evaluadora, Directorio CIPFIE, Tribunal Asesor, C.D.	

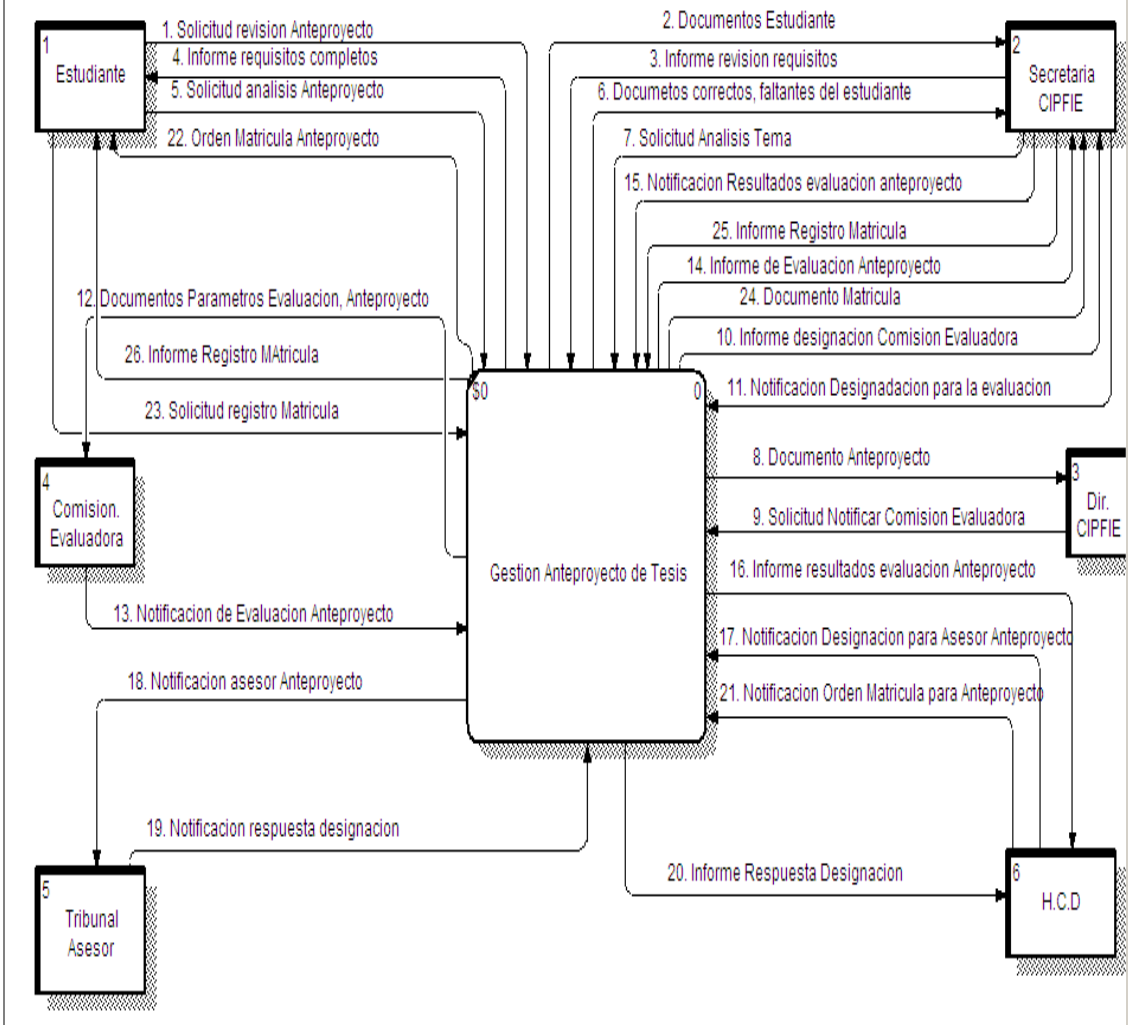


Figura V.15.- "P**roceso:** Gestión Anteproyecto de tesis"

PROCESO: GESTION PROYECTO DE TESIS - DIAGRAMA INTEGRADO

PROCESO: GESTIÓN PROYECTO DE TESIS	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE, Comisión Evaluadora, Directorio CIPFIE, Tribunal Asesor, C.D.	

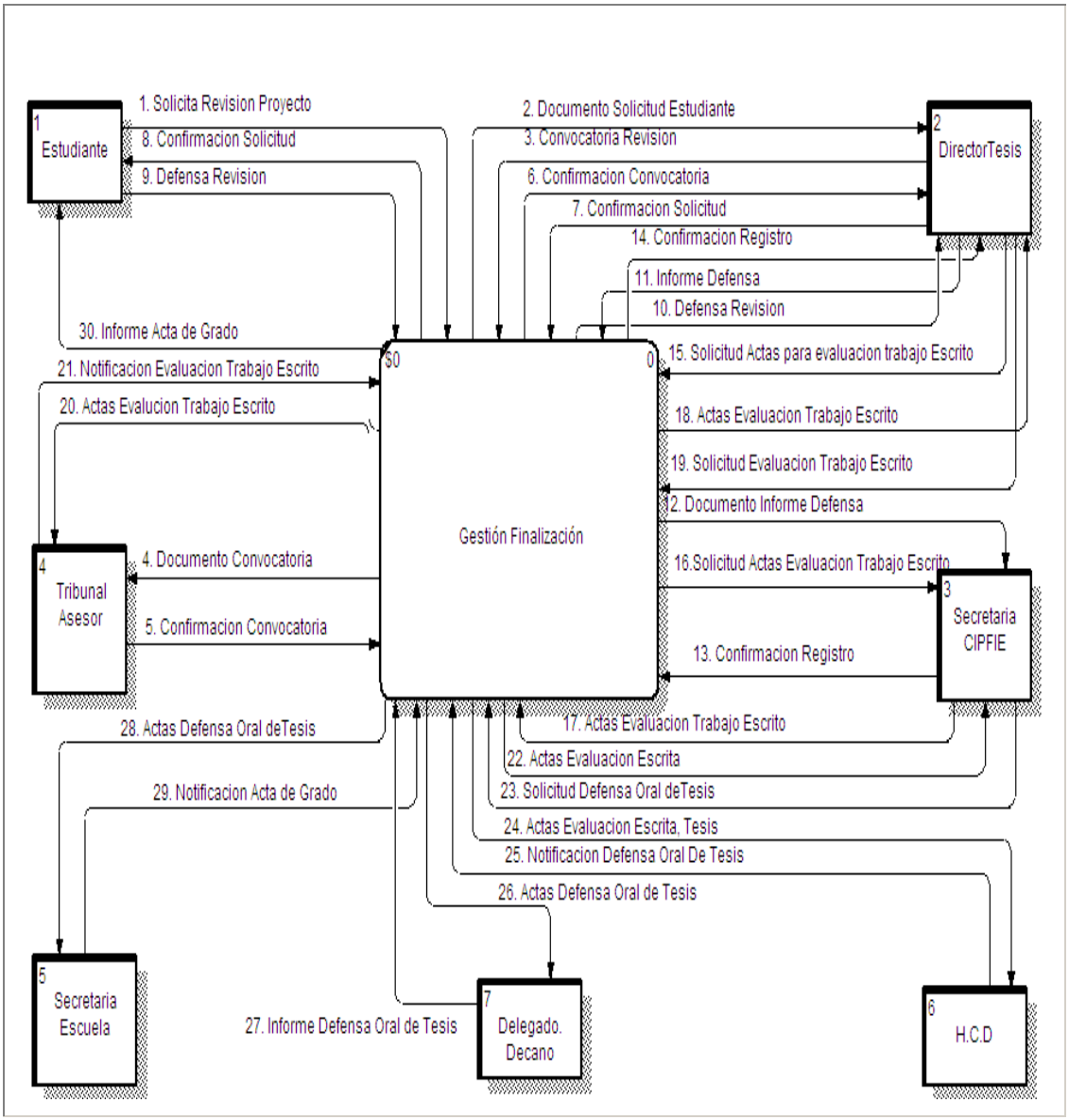


Figura V.16.- "P**roceso:** Gestión Proyecto de tesis"

5.2.2.3.1.3 DIAGRAMAS DE ACTIVIDAD

PROCESO: GESTION ANTEPROYECTO DE TESIS – DIAGRAMAS INDIVIDUALES

PROCESO: GESTIÓN ANTEPROYECTO DE TESIS **SUBPROCESO:** GESTION FACTIBILIDAD	**AUTOR:** JPT/LAR. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE, Comisión Evaluadora, Directorio CIPFIE	

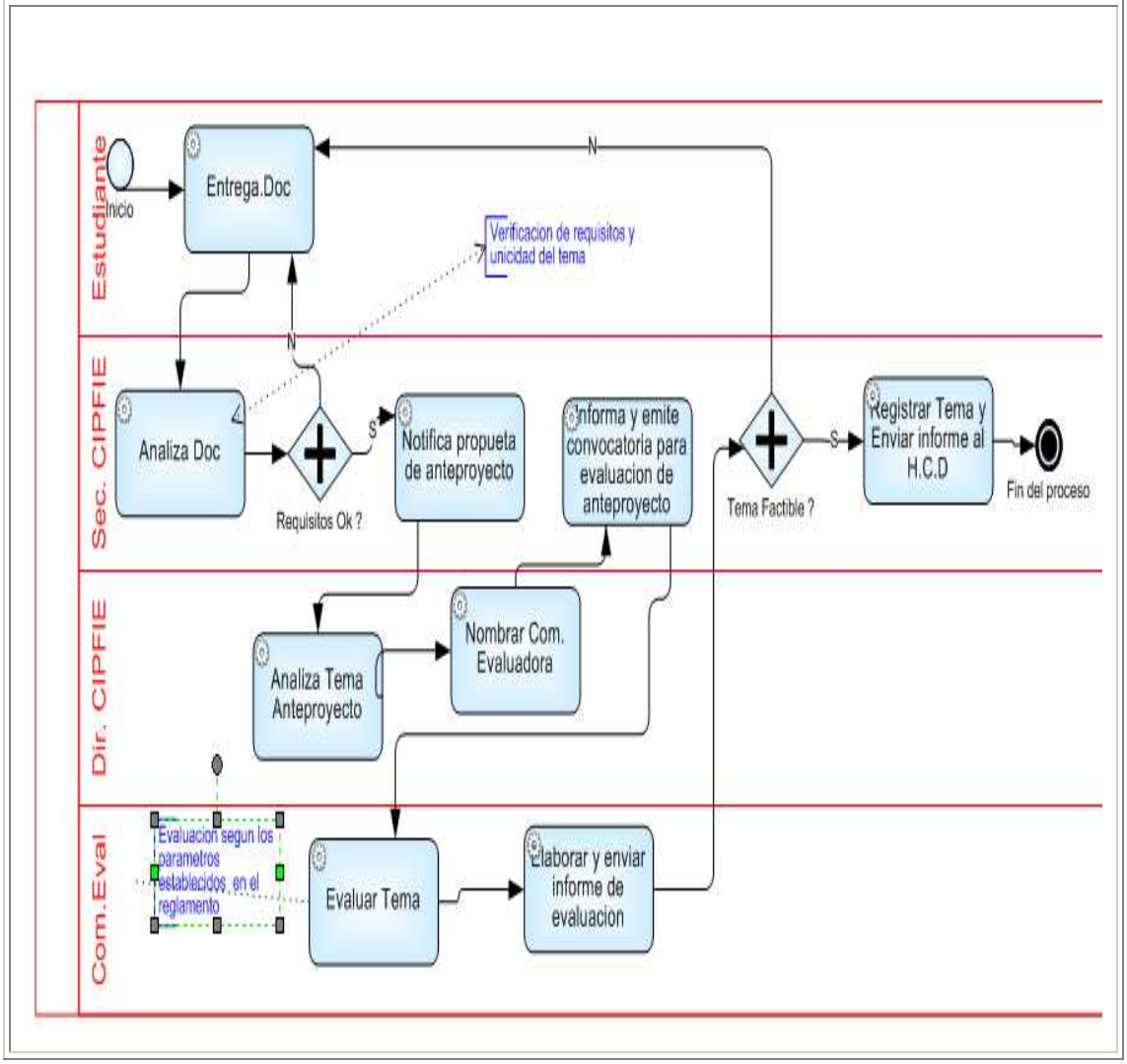


Figura V.17.- "**Diagrama de Actividad:** Subproceso Gestión Factibilidad"

PROCESO: GESTIÓN ANTEPROYECTO DE TESIS **SUBPROCESO:** GESTION LEGAL	**AUTOR:** JPT/LAR. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE, C.D, Asesor Tesis	

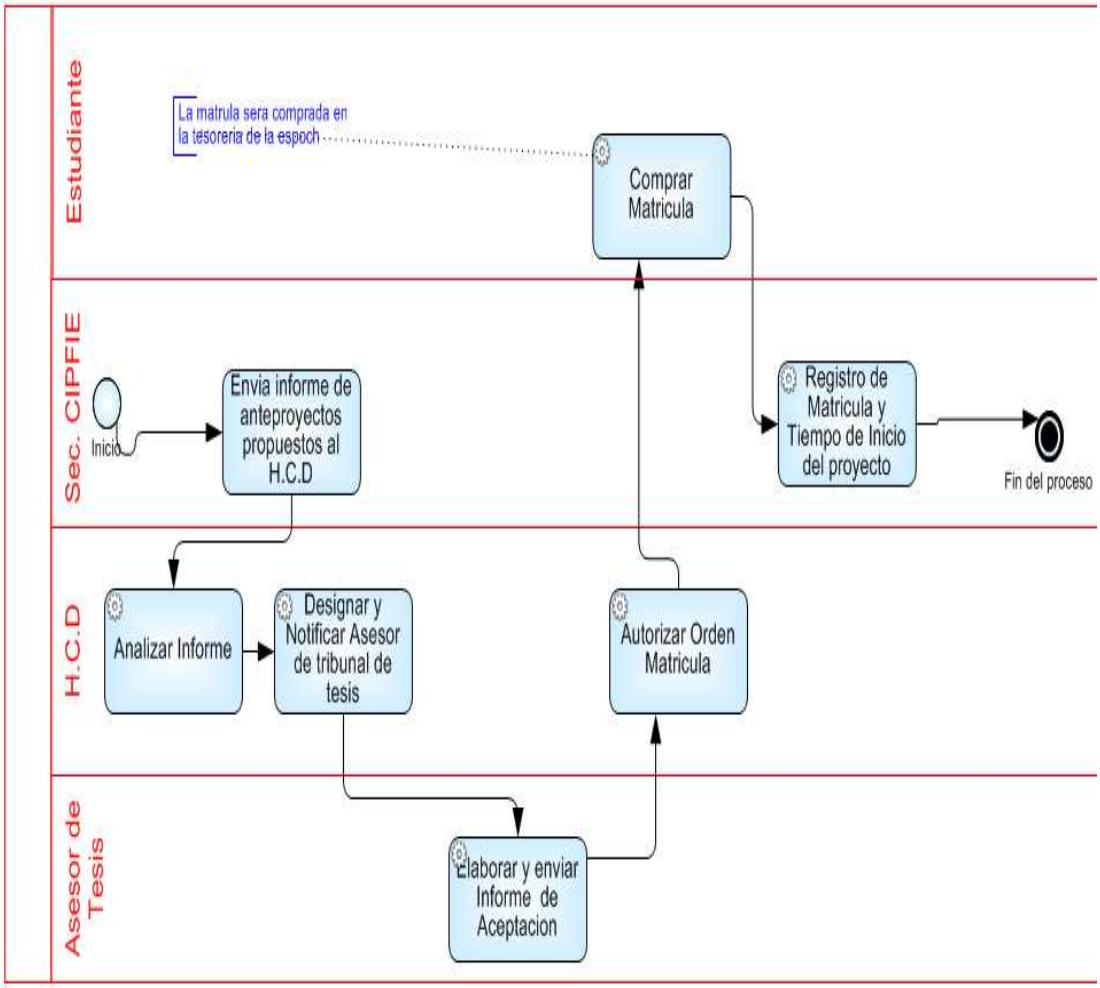


Figura V.18.- "**Diagrama de Actividad:** Subproceso: Gestión Legal**"**

PROCESO: GESTION PROYECTO DE TESIS – DIAGRAMAS INDIVIDUALES

PROCESO: GESTIÓN PROYECTO DE TESIS **SUBPROCESO:** GESTION MATRICULA	**AUTOR:** JPT/LAR. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE, C.D	

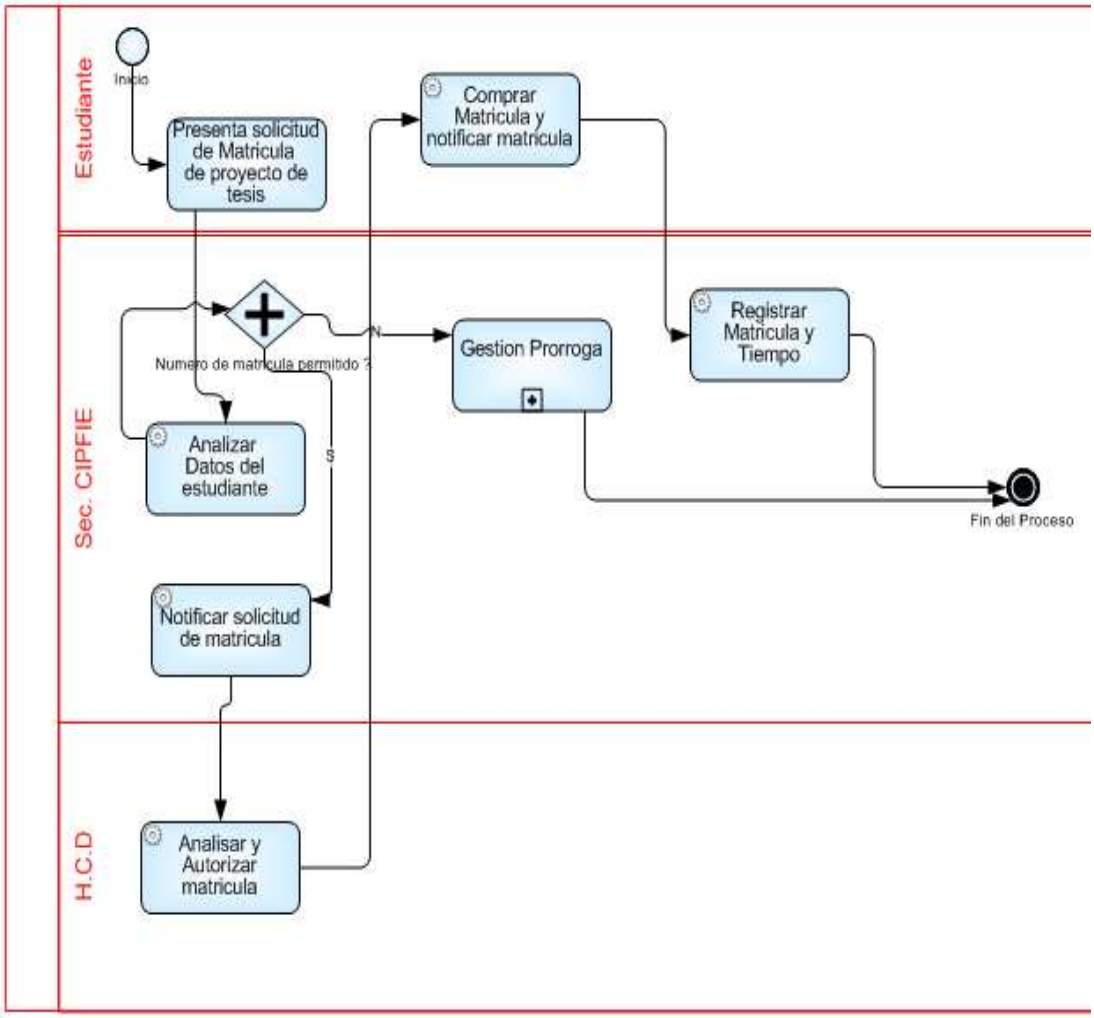


Figura V.19.-"**Diagrama de Actividad**: Subproceso: Gestión Matricula"

PROCESO: GESTIÓN PROYECTO DE TESIS **SUBPROCESO:** GESTION MIEMBROS	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE, C.D	

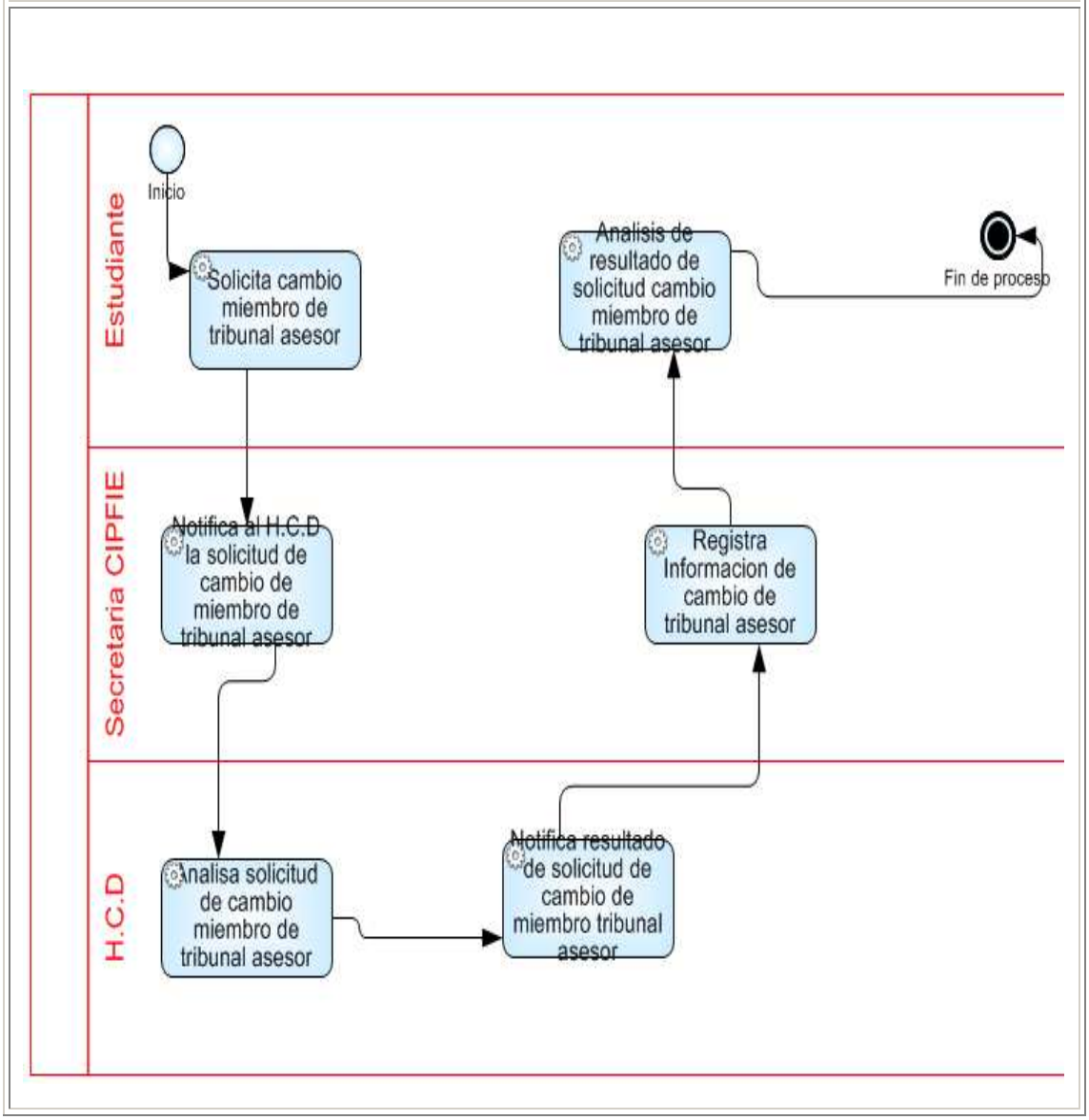


Figura V.20.-"**Diagrama de Actividad:** Subproceso Gestión Miembros"

PROCESO: GESTIÓN PROYECTO DE TESIS **SUBPROCESO:** GESTION ANULACION	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES Director Tesis, Secretaria CIPFIE, Directorio CIPFIE	

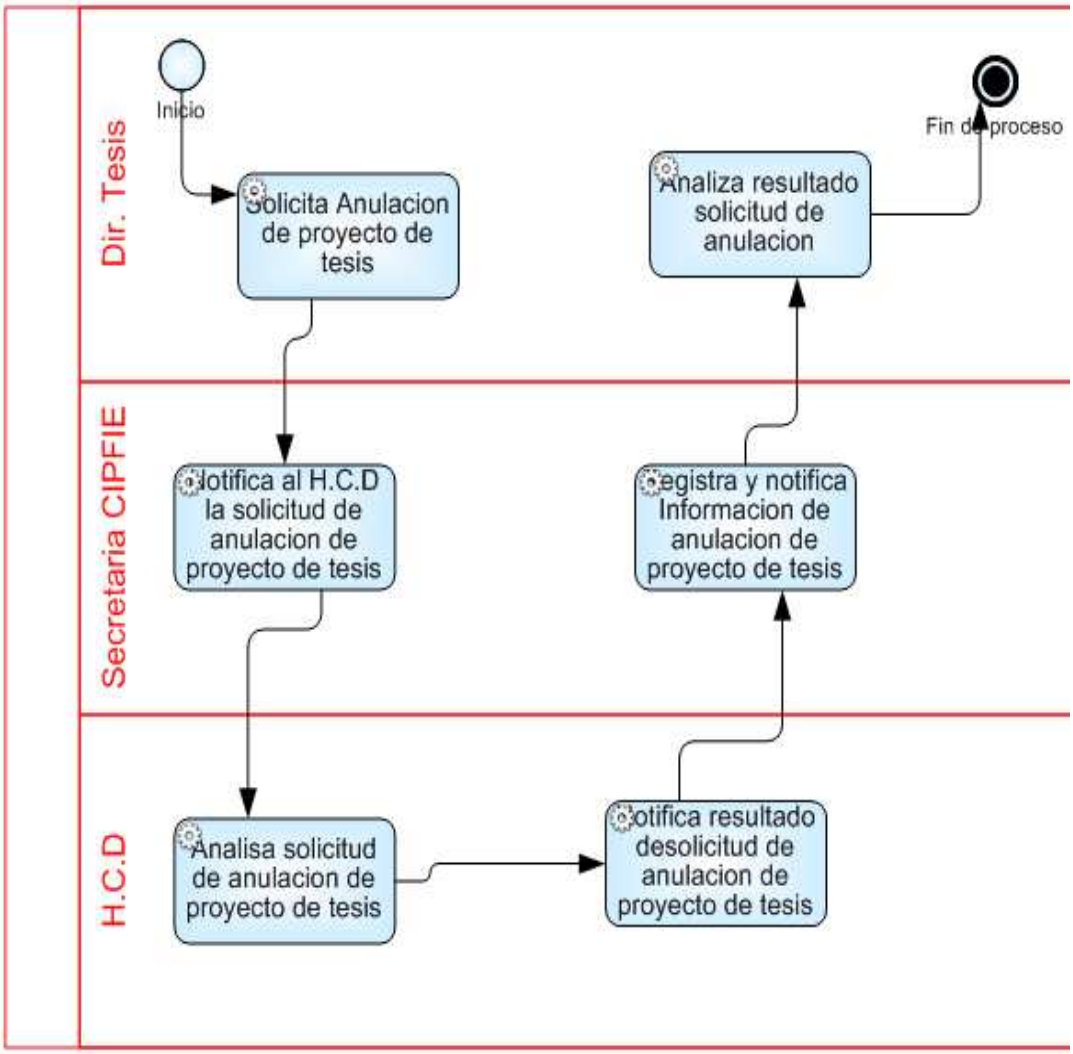


Figura V.21.-"**Diagrama de Actividad:** Subproceso Gestión Anulación"

PROCESO: GESTIÓN PROYECTO DE TESIS **SUBPROCESO:** GESTION FINALIZACION	**AUTOR:** Luis R. **FECHA:** 12 de Marzo del 2006
ACTORES Estudiante, Director Tesis, Secretaria CIPFIE, Tribunal Asesor, C.D, Secretaria Escuela, Delegado Decano de la Facultad.	

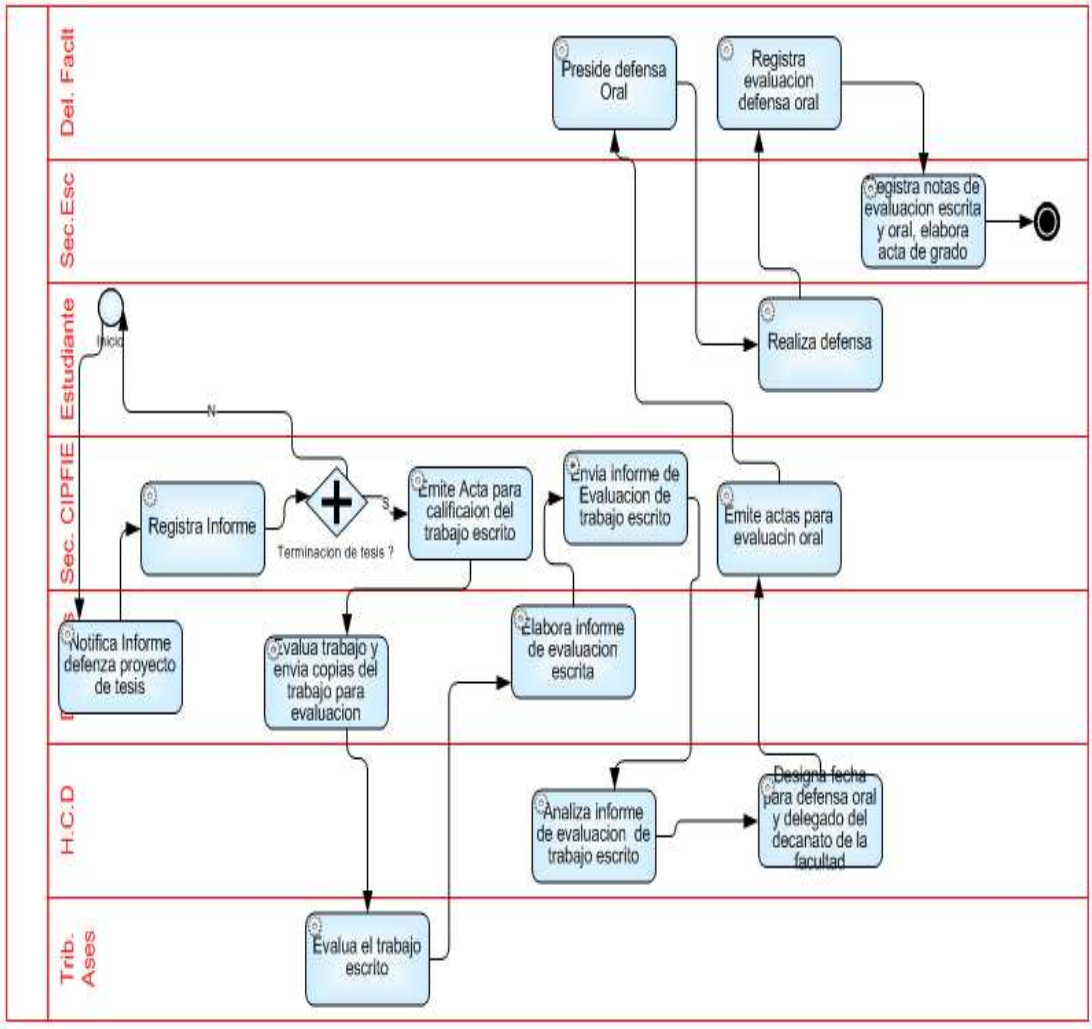


Figura V.22.-"**Diagrama de Actividad:** Subproceso Gestión Finalización"

PROCESO: : GESTIÓN PROYECTO DE TESIS **SUBPROCESO:** GESTION PRORROGA	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES Director de tesis, Secretaria CIPFIE, C.D.	

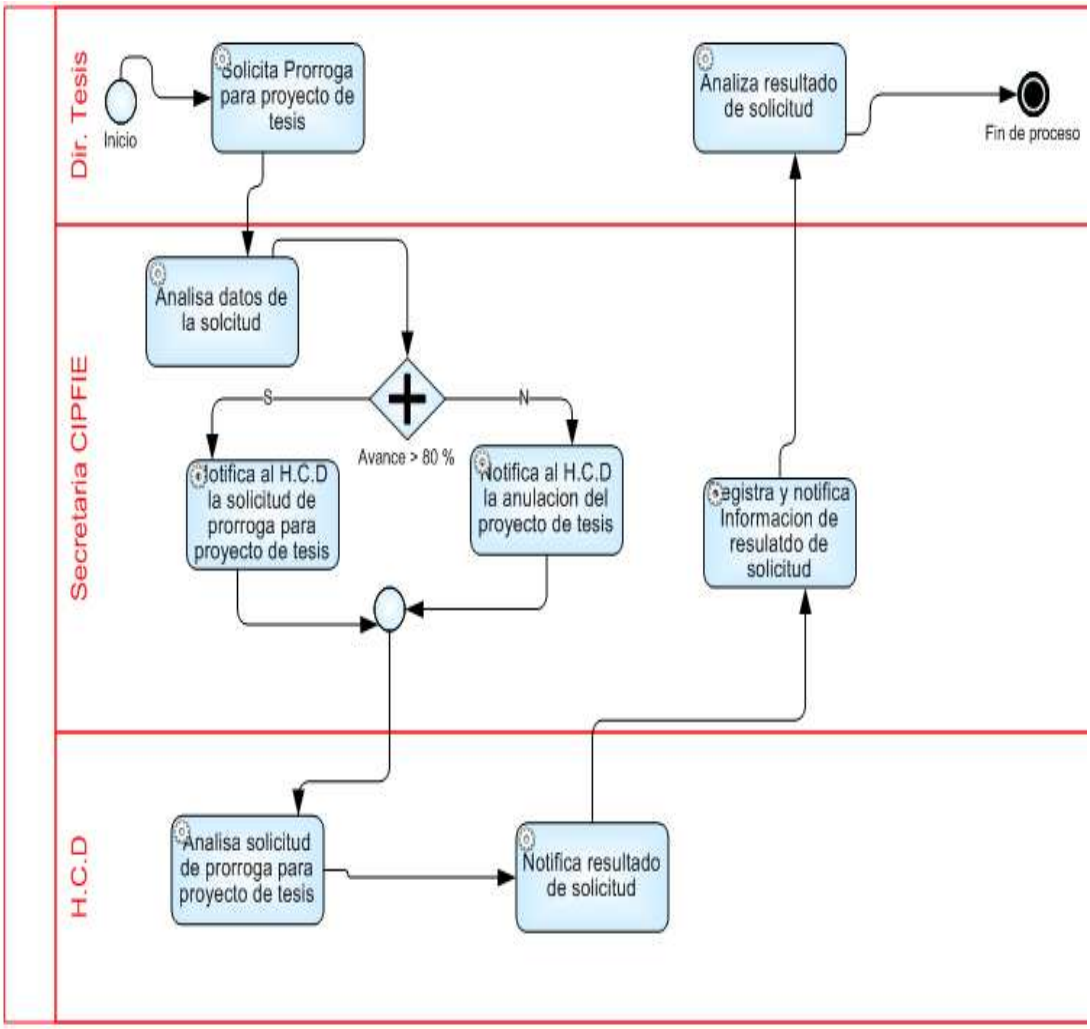


Figura V.23.-"**Diagrama de Actividad:** Subproceso Gestión Prorroga"

PROCESO: GESTION INFORMACION – DIAGRAMAS INDIVIDUALES

PROCESO: GESTIÓN INFORMACION **SUBPROCESO:** ANTEPROYECTOS DE TESIS	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE	

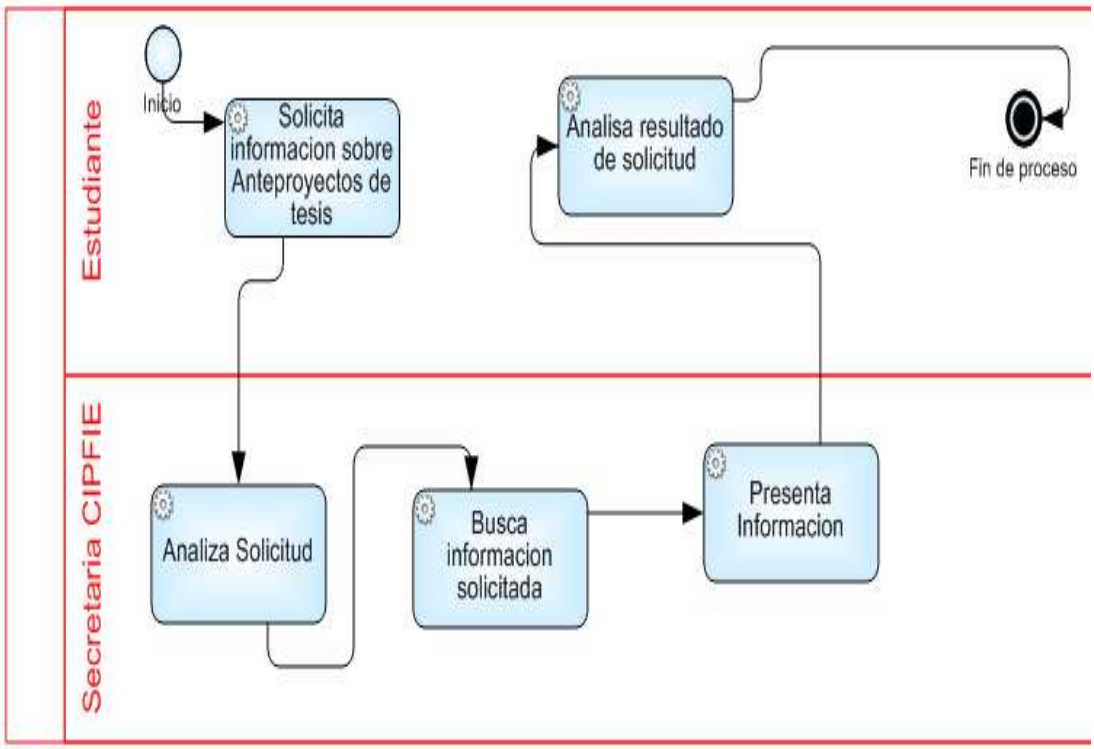


Figura V.24.-"**Diagrama de Actividad:** Subproceso Anteproyectos de tesis"

PROCESO: GESTIÓN INFORMACION **SUBPROCESO:** PROYECTOS DE TESIS	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE	

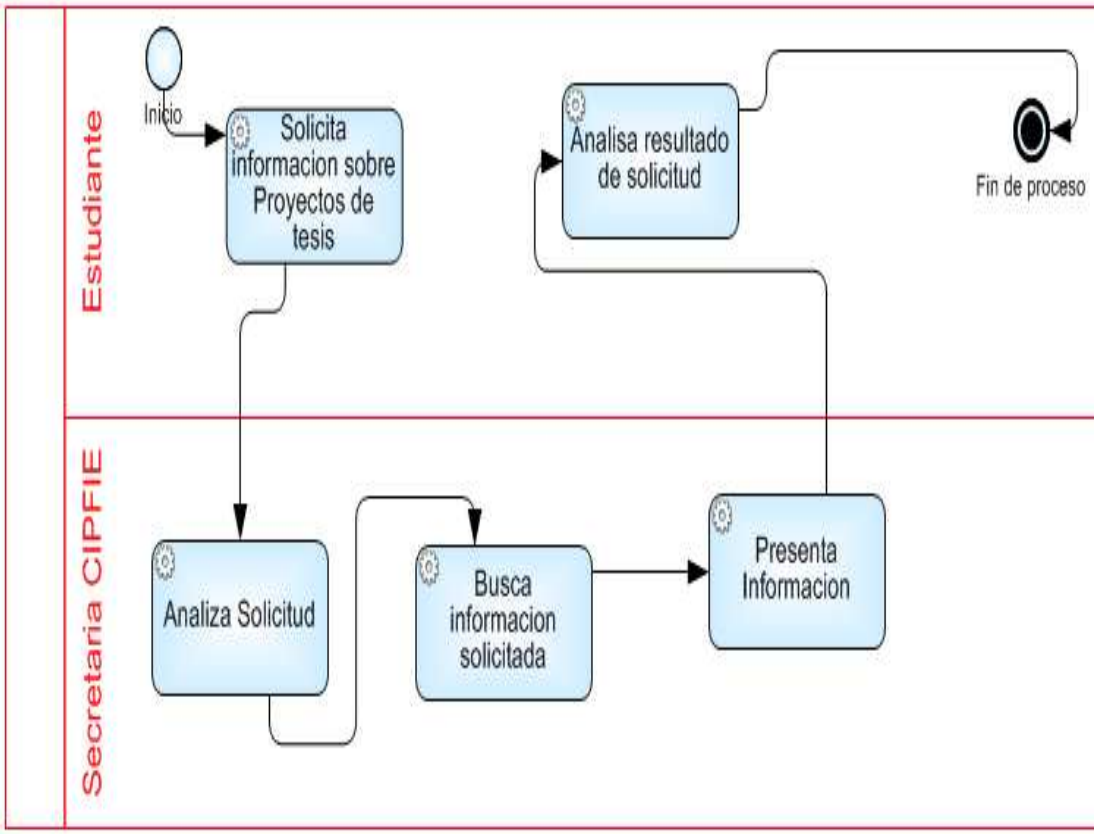


Figura V.25.-"**Diagrama de Actividad:** Proyectos de Tesis"

PROCESO: GESTIÓN INFORMACION **SUBPROCESO:** DIRECTORES DE TESIS	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE	

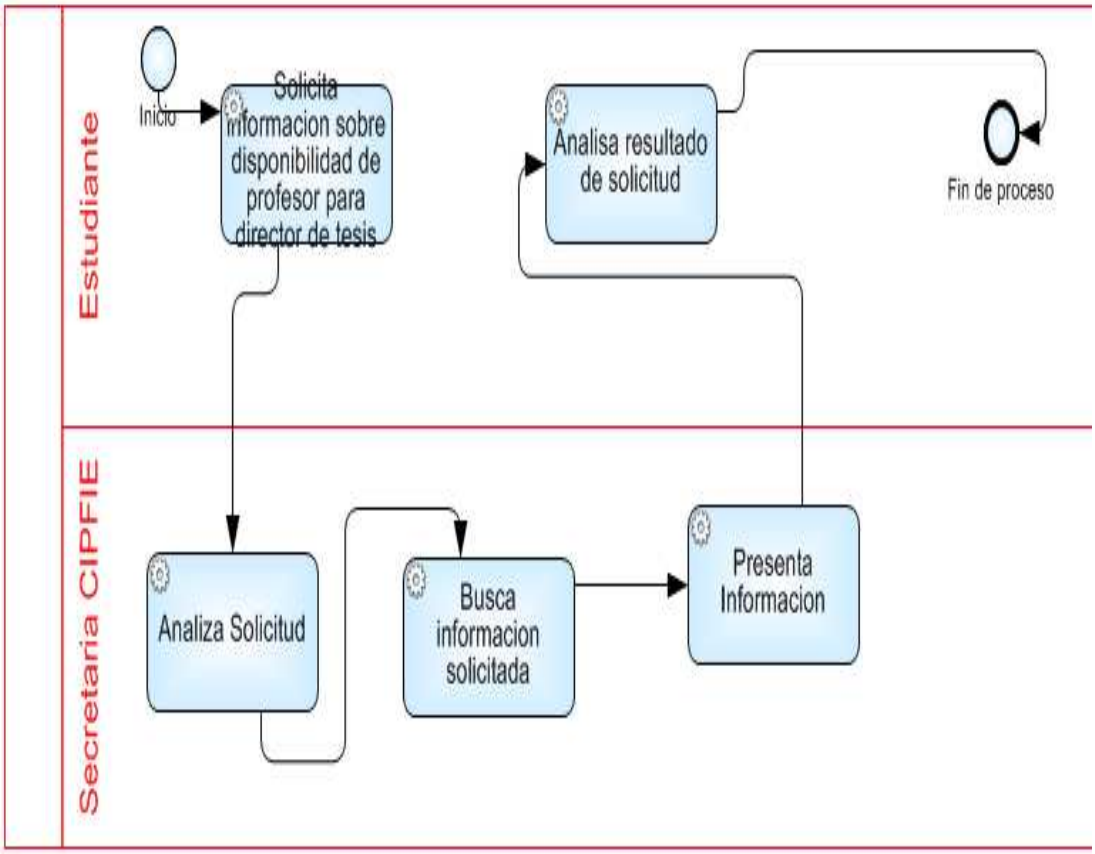


Figura V.26.-"**Diagrama de Actividad:** Directores de tesis"

PROCESO: GESTION ANTEPROYECTO DE TESIS - DIAGRAMA INTEGRADO

PROCESO: GESTIÓN ANTEPROYECTO DE TESIS	**AUTOR:** Luis R. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE, Comisión Evaluadora, Directorio CIPFIE, Tribunal Asesor, C.D.	

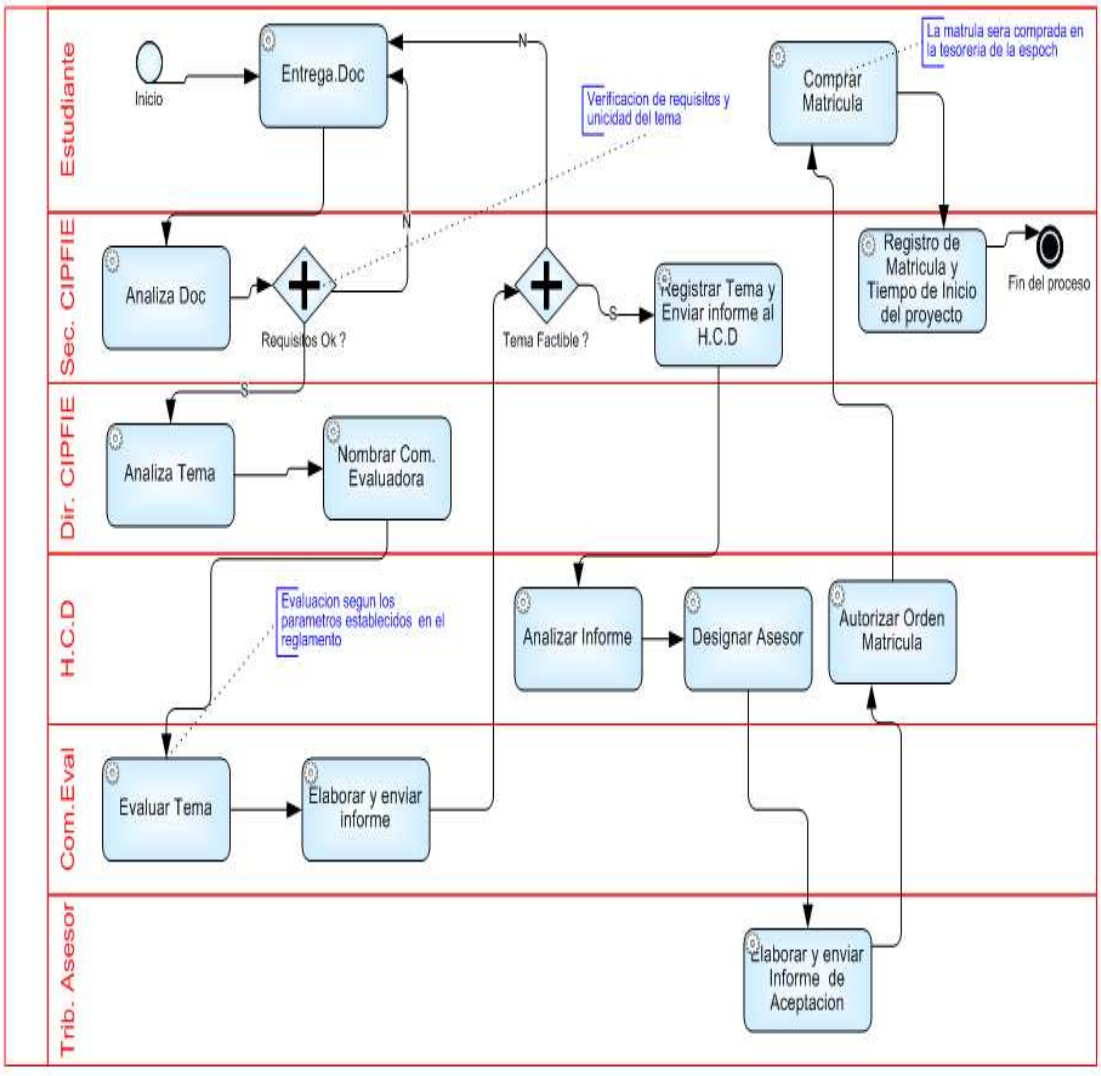


Figura V.27.-"**Diagrama de Actividad:** Proceso Anteproyectos de tesis"

PROCESO: GESTION PROYECTO DE TESIS - DIAGRAMA INTEGRADO

PROCESO: GESTIÓN PROYECTO DE TESIS	**AUTOR:** Luís R. **FECHA: 1**2 de Marzo del 2006
ACTORES Estudiante, Secretaria CIPFIE, Comisión Evaluadora, Directorio CIPFIE, Tribunal Asesor, C.D.	

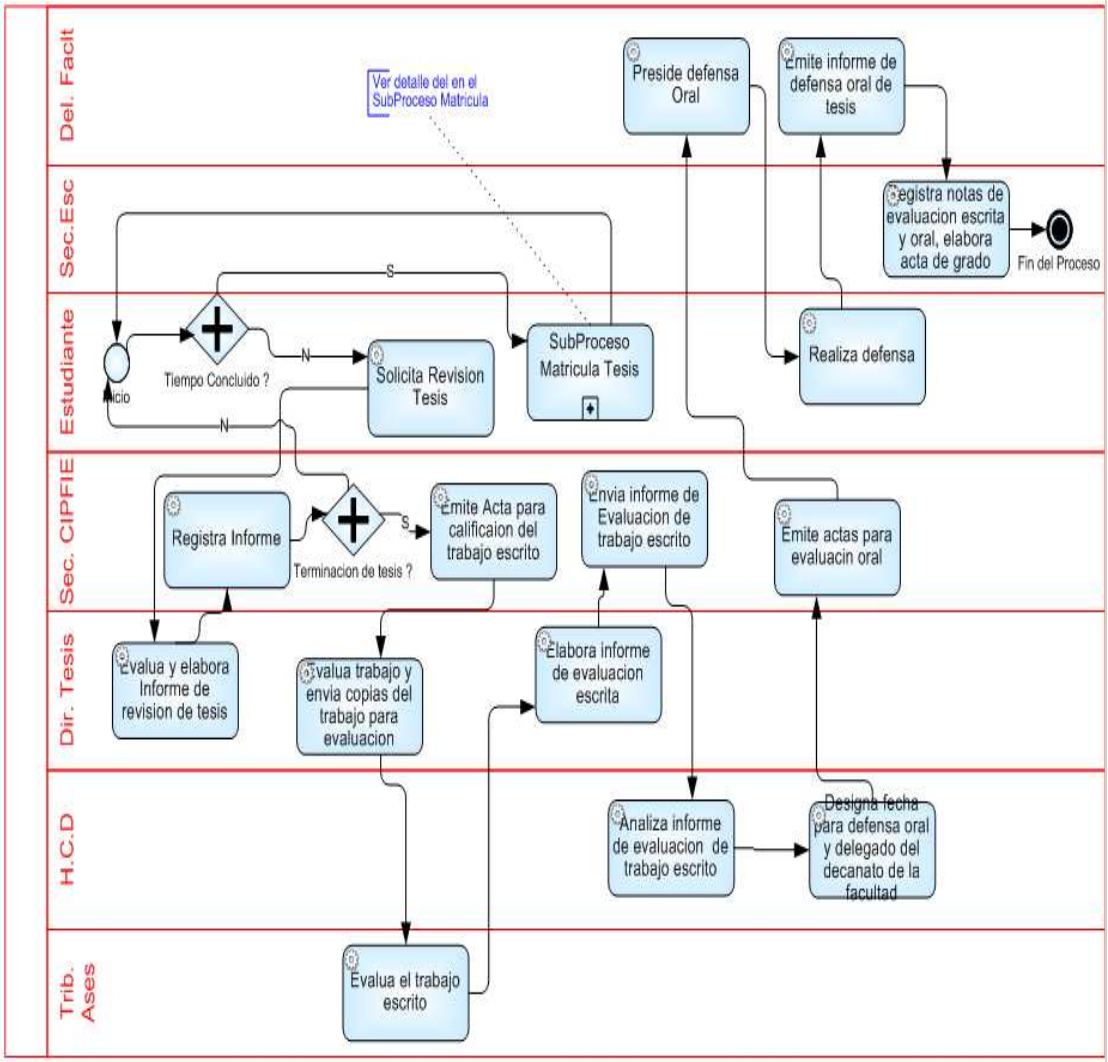


Figura V.28.-"**Diagrama de Actividad:** Proceso Gestión Proyectos de tesis"

ALTERNATIVAS DE SOLUCIÓN

Tabla V.19. Problemas soluciones detectados

PROCESO	PROBLEMAS DETECTADOS	SOLUCION
GESTION FACTIBILIDAD	• Investigaciones repetidas, existen muchos temas parecidos por no existir un control adecuado en la validación del tema de proyecto de tesis. • Excesivo tiempo en la clasificación y registro de los proyectos de tesis.	• Centralizar la información en un servidor de base de datos. • Por medio de un sistema informático verificar y registrar los temas de los proyectos a desarrollar.
GESTION MATRICULA	• Falta de control de los tiempos de inicio, fin y prorrogas de los proyectos de tesis, existiendo proyectos de tesis con tiempos de desarrollo que exceden los permitidos por el centro de investigación.	• Por medio de un sistema informático registrar los tiempos de inicio y fin de los proyectos de tesis logrando tener un control automático del tiempo de desarrollo.
GESTION FINALIZACION	• Procedimientos tediosos y manuales. • No existe un control sobre las evaluaciones obtenidas por los estudiantes en cada uno de los proyectos de tesis desarrollados. • Excesivos tiempos en la codificación de las tesis una ves que han sido terminadas.	• Con la implementación de un sistema informático automatizar tareas que demandan demasiado tiempo. • Registro de evaluaciones de los proyectos de tesis. • Generación de convocatorias, actas de evaluación automática. • Generación de códigos automáticos de las tesis una vez culminadas.

PROCESO	PROBLEMAS DETECTADOS	SOLUCION UTILIZAR TECNOLOGIA
GESTION INFORMACION ANTEPROYECTOS TESIS	• Falta de mecanismos de difusión. • Tiempos altos en búsqueda de la información. • Inseguridad en la información.	• Por medio de un sistema informático reducir el tiempo de búsquedas para obtener la información de proyectos de tesis. • Protección de la información por medio de la restricción de acceso al sistema.
GESTION INFORMACION PROYECTOS TESIS	• Falta de promoción de proyectos realizados y en realización. • Información almacenada en archivos físicos, sujeta a perdidas. • Inseguridad en la información.	• Por medio de la creación de una pagina Web, difundir los proyectos desarrollados. • Por medio de un sistema informático reducir el tiempo de búsquedas para obtener la información de proyectos de tesis. • Protección de la información por medio de la restricción de acceso al sistema.

ESPECIFICACIÓN DE REQUISITOS SOFTWARE (Anexo).

ESTUDIO DE FACTIBILIDAD

5.2.2.4 FACTIBILIDAD OPERATIVA

Para determinar la factibilidad operativa del sistema a desarrollar nos apoyaremos en el comportamiento del sistema actual, permitiendo determinar el comportamiento que tendrá el sistema a desarrollar.

- **Formulación del Problema.**

El problema en forma general se presenta como la falta de funcionalidad y control en los procesos para la gestión de anteproyectos y proyectos de tesis, dichos procesos en la actualidad se efectúan en forma manual, ocasionando deficiencias en el manejo y manipulación de información. Dentro de los problemas a solucionar se destaca los siguientes:

- **Gestión de Administración del sistema actual y propuesto del CIPFIE.**

Tabla V.20. – Gestión de administración del sistema actual y propuesto del CIPFIE.

Actividad	Sistema Actual	Sistema Propuesto
Validación unicidad tema proyecto de tesis	20 minutos	2 minutos
Validación Cupo Profesor Director Tesis	15 minutos	2 minutos
Ingreso de información de Anteproyectos de tesis	10 minutos	3 minutos
Generación de Convocatorias para Evaluación de Anteproyectos	10 minutos	2 minutos

Generación de Informes	20 minutos	3 minutos
Ingreso de Información de Documentos de un proyecto(Resoluciones)	10 minutos	2 minutos
Generación de Reportes sobre información de Anteproyectos	20 minutos	1 minuto
Generación de Reportes sobre información de Proyectos	20 minutos	1 minuto
Generación de Actas de evaluación Escrita	5 minutos	1 minuto
Generación de Actas de evaluación Oral	5 minutos	1 minuto
Generación de Reportes Estadísticos	2 h	2 minutos
TIEMPO TOTAL	**255 minutos**	**20 minutos**

De acuerdo a la tabla Tabla V.20 , se puede observar claramente que el tiempo de respuesta del Sistema Actual es muy diferente frente al Sistema Propuesto en un gran porcentaje por lo que se puede decir que la automatización de dicha entidad va ha ser en un 93% operativamente eficiente y de gran ayuda para los que laboran.

- **Oportunidades de Mejoría**

Las posibles características de mejora en el desarrollo del sistema para la gestión de los anteproyectos y proyectos de tesis del CIPFIE se respaldan en los siguientes aspectos:

  - Incremento en la velocidad de ingreso y procesamiento de la información de los anteproyectos y proyectos de tesis
  - Agilización de un proceso mediante la eliminación de pasos innecesarios o duplicados

- Reducción de errores en la entrada de datos por medio de cambios en formas y pantallas
- Reducción de salida Redundante
- Posibilidad de agregar grandes cantidades de datos de distintas formas que sean útiles para la planificación y la toma de decisiones
- Mejores posibilidades de mantener una continua monitorización de los procesos e información.
- Reducción de la necesidad de trabajo forzado en el control de procesos y de recursos.
- Mejora en la seguridad, almacenamiento y portabilidad de registros.

5.2.2.5 FACTIBILIDAD TECNICA

A continuación se realiza la caracterización con respecto al hardware, software y recursos humanos y adicionales que demanda la nueva aplicación con la finalidad de convertir al proyecto en algo técnicamente factible.

DATOS ACTUALES DISPONIBLES

- **IDENTIFICACIÓN DEL SISTEMA**

  TITULO: Sistema de Administración de Proyectos de la Facultad de Informática y Electrónica.

  NOMBRE CARACTERISTICO: SIGEP.

- **RECURSOS HUMANOS INVOLUCRADOS**

  NUMERO DE DESARROLLADORES: Dos.

  NUMERO DE ASESORES: Dos.

  NUMERO DE USUARIOS: Dos.

- **RECURSOS HARDWARE**

NUMERO DE COMPUTADORAS DE DESARROLLO: Dos.

NUMERO DE ESTACIONES DE TRABAJO: Dos.

NUMERO DE DISPOSITIVOS DE IMPRESIÓN: Uno.

- **RECURSOS SOFTWARE**

SISTEMA OPERATIVO: Windows XP.

SISTEMA ACTUAL: No existe ningún sistema automatizado.

- **GRUPO DE TRABAJO.**

Las personas que intervienen en el desarrollo del sistema de automatización son las siguientes.

DIRECTOR O COORDINADOR: Ing. Danilo Pastor.

VINCULADOS:

Ing. Danilo Pastor (Director del CIPFIE).

Lcda. Maria Agusta Larrea (Secretaria del CIPFIE).

INFORMATICOS:

Juan Pablo Tixe.

Luís Rivera.

5.2.2.5.1 RECURSOS HARDWARE PARA DESARROLLO

La tabla V.21. describe el recurso hardware para el desarrollo del proyecto

Tabla V.21. –Recurso hardware para el desarrollo del sistema SIGEP.

Nº	Características	S. Operativo	Objetivo	Responsable
1	Computador Pentium 4. Disco de 70 GB. Memoria Ram 256 Mgb.	Microsoft Windows XP Profesional	Documentación Programación	Luis Rivera
2	Computador Pentium 4.	Microsoft Windows XP Profesional	Documentación Programación	Juan Pablo Tixe

	Disco de 70 GB. Memoria Ram 256 Mgb.			

5.2.2.5.2 RECURSO HARDWARE PARA LA IMPLANTACION

La tabla V.22 describe el recurso hardware para la implementación

Tabla V.22. –Recurso hardware para la implantación del sistema SIGEP.

Nº	Características	Objetivo	Responsable
1	Computador Pentium 4. Disco de 70 GB. Memoria Ram 512 Mgb.	Servidor de Aplicaciones	ESPOCH
2	Computador Pentium 4. Disco de 70 GB. Memoria Ram 512 Mgb.	Servidor de Base de Datos	ESPOCH
3	Computador Pentium 4. Disco de 70 GB. Memoria Ram 256 Mgb.	Cliente	ESPOCH

5.2.2.5.3 RECURSOS SOFTWARE PARA DESARROLLO.

La tabla V.23 describe el recurso software para el desarrollo del proyecto

Tabla V.23. –Recurso software para el desarrollo del sistema SIGEP.

Nº	Recurso	Utilización
1	Logic Works BPWin 2.03	Modelado de Procesos
2		Representación de Procesos con notación

	Process Modeler 4.0	BPMN
3	Microsoft Office 2003	Documentación
4	Microsoft VisualStudio.net 2005	Desarrollo
5	Microsoft SQLServer 2000	Motor de Base de datos
6	Microsoft Office Project 2003	Planificación y Gestión
7	Microsoft Windows XP Profesional	Sistema Operativo

5.2.2.5.4 RECURSOS SOFTWARE PARA LA IMPLANTACION

La tabla V.24 describe el recurso software para la implantación del proyecto

Tabla V.24. –Recurso software para la implantación del sistema SIGEP.

Nº	Recurso	Utilización
1	Windows 2003 Server	Servidor de Aplicaciones
2	Microsoft SQLServer 2000	Servidor de Base de Datos
3	Microsoft .net Framework 2.0, 3.0	Entorno de Ejecución para la aplicación

Tomando en cuenta las descripciones anteriores del hardware y software para el desarrollo e implantación, podemos concluir que la ESPOCH y por consiguiente el CIPFIE posee la tecnología apropiada para convertir al proyecto técnicamente factible.

5.2.2.6 FACTIBILIDAD ECONOMICA

5.2.2.6.1 COSTOS TANGIBLES DEL PROYECTO

Tabla V.25. – Costos de Recursos Software.

CANTIDAD	DESCRIPCIÓN	VALOR UNITARIO	VALOR TOTAL	FUENTE DE FINANCIAMIENTO
Licenciamiento				
1	Microsoft Framework 2.0	----	----	LIBRE
1	SQL Server 2000	$750	$750	ESPOCH
10	Microsoft Office 2003 Professional.	$150	$1500	ESPOCH
10	Windows Xp Professional SP2.	$80	$800	ESPOCH
1	Windows 2003 Server SP1.	$1000	$1000	ESPOCH
TOTAL LICENCIAMIENTO			**$5050,00**	
COSTO DE LA CAPACITACION DEL SISTEMA			**$00,00**	
Desarrollo de Software			----	
2	Sistema de administración de proyectos de la facultad de Informática y Electrónica	$0	$0	TESISTAS
TOTAL DESARROLLO DE SOFTWARE			$0	
TOTAL PRESUPUESTO			**$5050,00**	

Cabe mencionar que la ESPOCH ya posee el licenciamiento para los recursos mencionados en la tabla Tabla V.25, por tal motivo podemos concluir que el proyecto es económicamente factible.

5.2.2.6.2 BENEFICIOS INTANGIBLES DEL PROYECTO

- Mejor Calidad del diseño mediante una optimización iterativa.

- Mayor satisfacción del cliente debido a un control programable y unas mejores decisiones administrativas a partir de datos con formato previamente analizados.
- Satisfacción Laboral.
- Competitividad Tecnológica.
- Mayor Seguridad de los Datos.

5.2 EXPLORACION

5.3.1 HISTORIAS DE USUARIO

Se han definido las siguientes historias de usuario:

Tabla V.26. – Historia de usuario: Gestión factibilidad proyecto.

	Descripción de proceso de negocio
Numero: 1	**Usuario** : Todos
Nombre proceso : Gestionar Factibilidad Proyecto Propuesto	
Prioridad en negocio : Alta	**Riesgo de desarrollo :** Medio
Puntos Estimados: 3	**Interacción asignada: 2**
Programador Responsable : Luis Rivera y Juan Pablo Tixe	
El aspirante inicia la interacción al presentar los respectivos documentos en el decanato de la facultad, el decano envía los respectivos documentos al Centro de Investigación, donde la secretaria revisa los mismos y procede a registrar en el SiGep nombre, escuela, carrera, área, lugar de realización, estudiantes participantes y un director, el Sistema valida nombre único de Proyecto y guarda la información recabada. Una comisión asignada por el Director del centro de investigación nombra un tribunal para la evaluación del Proyecto Propuesto. La Secretaria procede a registrar el tribunal designado en el SiGEP y procede a Imprimir las Actas para la respectiva evaluación. Una vez que el Proyecto ha sido validado, la Secretaria registra en el SiGEP la factibilidad y las respectivas observaciones del Proyecto. Una vez que el proyecto ha sido aceptado se encuentra en Estado de Anteproyecto, para lo cual la Secretaria debe asignar un Asesor al Proyecto, ingresando información como numero y fecha de resolución.	
Observaciones: Si el tema no es único se anulara el tema. Si en la evaluación del Proyecto Propuesto el tema no logra una calificación mínima de 14, el tema no será factible.	

Tabla V.27. – Historia de usuario: Gestión matricula proyecto.

Descripción de proceso de negocio	
Numero: 2	**Usuario** : Todos
Nombre proceso : Gestionar Matricula Proyecto	
Prioridad en negocio : Alta	**Riesgo de desarrollo :** Medio
Puntos Estimados: 3	**Interacción asignada: 2**
Programador Responsable : Luis Rivera y Juan Pablo Tixe	
Una vez que el Proyecto ha sido validado y posee un Asesor, la Secretaria ingresa al SiGep y procede a Matricular el Proyecto para lo cual registra información como el Número de Comprobante de Pago, Fecha de Resolución, Fecha Final de Matricula; el sistema valida fecha de Matriculas vigentes, guarda y emite un mensaje con el resultado del proceso.	
Observaciones: Si existe una matrícula con fecha vigente no será factible registrar otra.	

Tabla V.28. – Historia de usuario: Gestión Avance Proyecto.

Descripción de proceso de negocio	
Numero: 3	**Usuario** : Todos
Nombre proceso : Gestionar Avance Proyecto	
Prioridad en negocio : Alta	**Riesgo de desarrollo :** Medio
Puntos Estimados: 3	**Interacción asignada: 2**
Programador Responsable : Luis Rivera y Juan Pablo Tixe	
Si el Proyecto posee una Matricula vigente válida, la Secretaria podrá registrar Avances para el Proyecto, ingresando Fecha de Avance y porcentaje obtenido. El Sistema contempla que la suma de porcentajes de los Avances no sea mayor a 100, así como que la fecha del Avance actual no sea menor a fechas de Avances existentes, la Secretaria guarda la información ingresada y el Sistema emite un mensaje del resultado del proceso.	
Observaciones: No se puede registrar dos Avance con la misma fecha.	

Tabla V.29. – Historia de usuario: Gestión Finalización tesis.

Descripción de proceso de negocio	
Numero: 4	**Usuario** : Todos
Nombre proceso : Gestionar Finalización de Tesis	
Prioridad en negocio : Alta	**Riesgo de desarrollo :** Medio
Puntos Estimados: 3	**Interacción asignada: 2**
Programador Responsable : Luis Rivera y Juan Pablo Tixe	
Una vez que el Proyecto ha cumplido con un porcentaje igual al 100% en Avances, está listo para la Defensa Final. Para lo cual la Secretaria utilizando el SiGEP asigna un Delegado ingresando el Nombre, fecha y número de resolución. Así mismo procede a imprimir las Actas para la respectiva evaluación oral y escrita. Posteriormente a la evaluación, el proyecto está listo para ser registrado como	

finalizado, para lo cual se ingresa información como el código de Tesis asignado, resumen, palabras claves, etc.
Observaciones: No se puede asignar un Delegado si el Proyecto no ha cumplido con el 100% en Avances.

Tabla V.30. – Historia de usuario: Gestión Emisión Actas.

	Descripción de proceso de negocio	
Numero: 5	**Usuario** : Todos	
Nombre proceso : Gestionar Emisión de Actas Calificación		
Prioridad en negocio : Alta	**Riesgo de desarrollo :** Medio	
Puntos Estimados: 3	**Interacción asignada: 2**	
Programador Responsable : Luis Rivera y Juan Pablo Tixe		
Con el fin evaluar el proyecto en sus diferentes Etapas es necesario imprimir Actas para dicho objetivo, para lo cual la Secretaria ingresa al SiGEP selecciona la opción "Documentos" en donde tiene la posibilidad de Imprimir Actas para Evaluación de Propuestos, Actas para Defensa Oral y Escrita.		
Observaciones: Únicamente se pueden imprimir Actas para Proyectos que cumplan con los requisitos establecidos en las reglas de negocio para este evento.		

Tabla V.31. – Historia de usuario: Gestión Consultas y reportes.

	Descripción de proceso de negocio	
Numero: 6	**Usuario** : Todos	
Nombre proceso : Consultas y Reportes		
Prioridad en negocio : Alta	**Riesgo de desarrollo :** Medio	
Puntos Estimados: 3	**Interacción asignada: 2**	
Programador Responsable : Luis Rivera y Juan Pablo Tixe		
La Secretaria tiene la posibilidad de consultar información de Proyectos; así como de generar reportes estadísticos relacionados a Empresas, Escuelas, Docentes, Estudiantes, con el fin de la correcta toma de decisiones a nivel Institucional.		
Observaciones:		

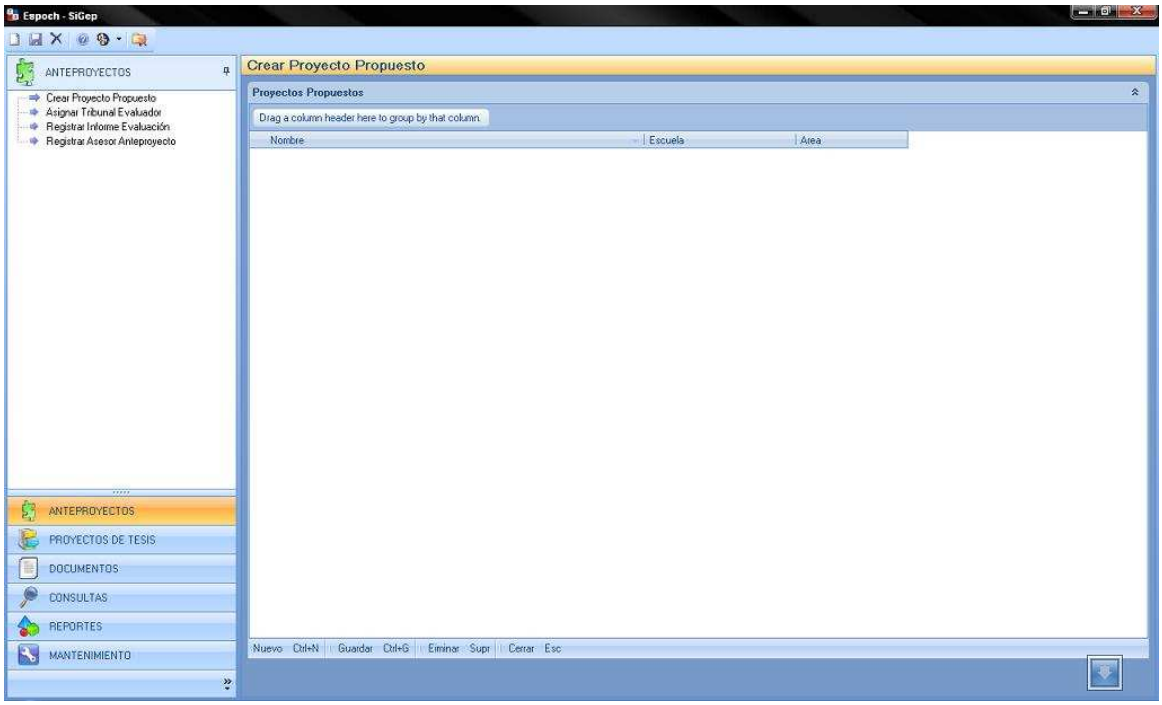
Tabla V.32. – Historia de usuario: Gestión Mantenimiento.

	Descripción de proceso de negocio
Numero:	**Usuario** : Todos

7	
Nombre proceso : Mantenimiento	
Prioridad en negocio : Alta	**Riesgo de desarrollo :** Medio
Puntos Estimados: 3	**Interacción asignada: 2**
Programador Responsable : Luis Rivera y Juan Pablo Tixe	
Cuando se requiera configurar parámetros del SiGEP, el usuario con el adecuado Rol, estará en la capacidad de personalizar dichos parámetros.	
Observaciones: Únicamente podrá configurar el SiGEP, el usuario con un Rol que contenga los permisos necesarios para realizar esta tarea.	

5.2.2 PROTOTIPO DE INTERFAZ

Se va a tomar el siguiente prototipo de interfaz como referencia :



5.3 PLANIFICACION DE LIBERACION

5.4.1. IDENTIFICACION DE SERVICIOS A DESARROLLAR

Las interfaces de servicio identificados en las historias de usuarios las podemos ver a continuación, estas interfaces de servicios serán desarrollados en cada una de las iteraciones en la siguiente fase:

Figura V.29.-"**Interface de servicio :** IAvanceServiceInterface"

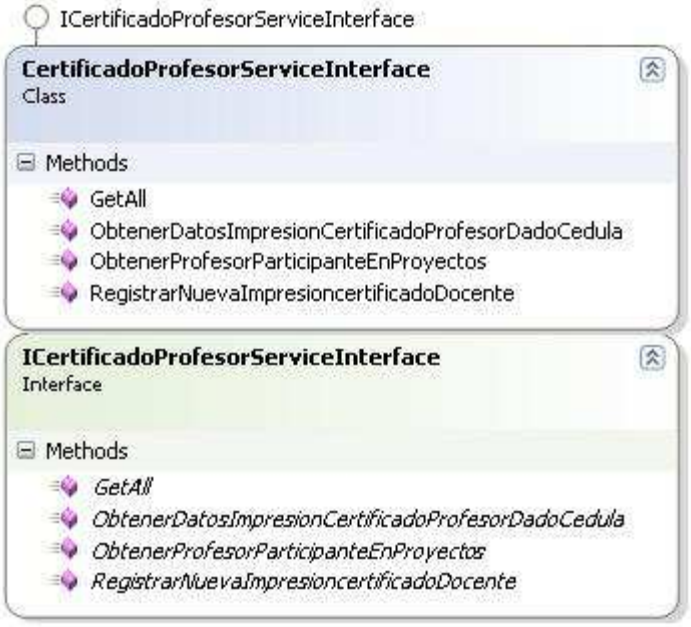


Figura V.30.-"**Interface de servicio:** ICertificadoProfesorServiceInterface"

Figura V.31.-"**Interface de servicio:** IDelegadoServiceInterface"

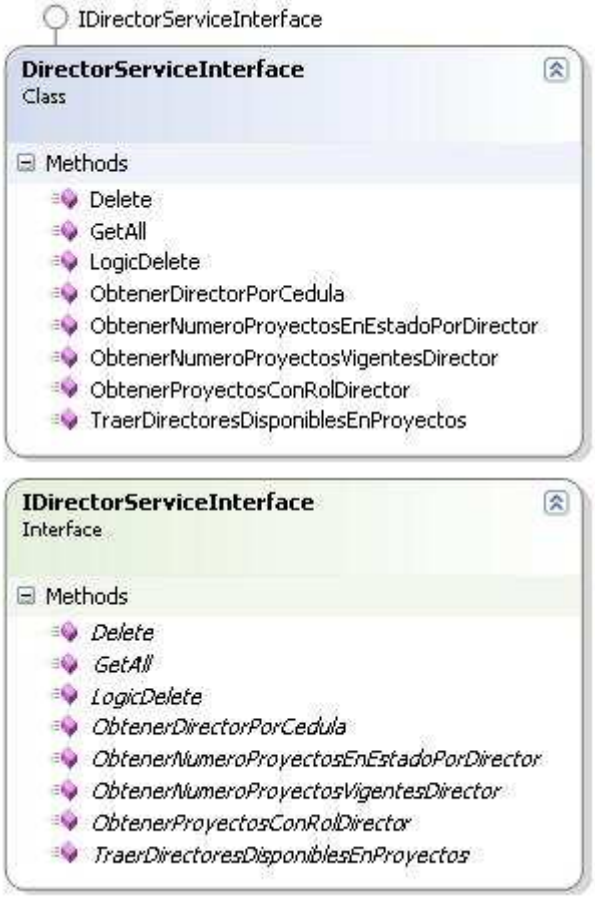


Figura V.32.-"**Interface de servicio:** IDirectorServiceInterface"

Figura V.33.-"**Interface de servicio:** IEmpresaServiceInterface"

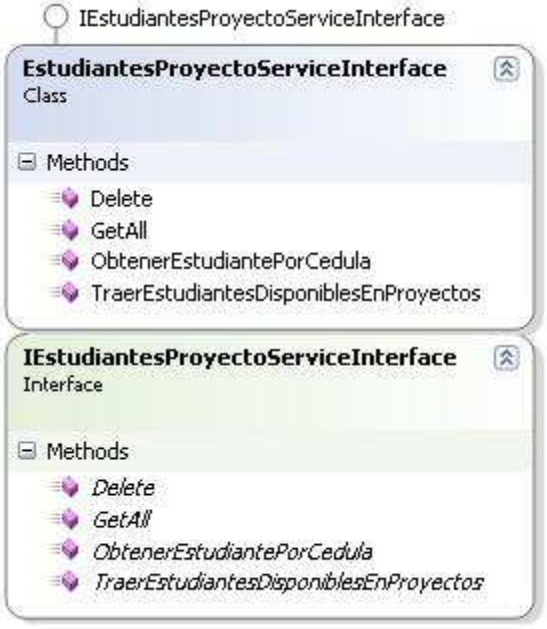


Figura V.34.-"**Interface de servicio:** IEstudianteProyectoServiceInterface"

Figura V.35.-"**Interface de servicio:** IEtapaServiceInterface"

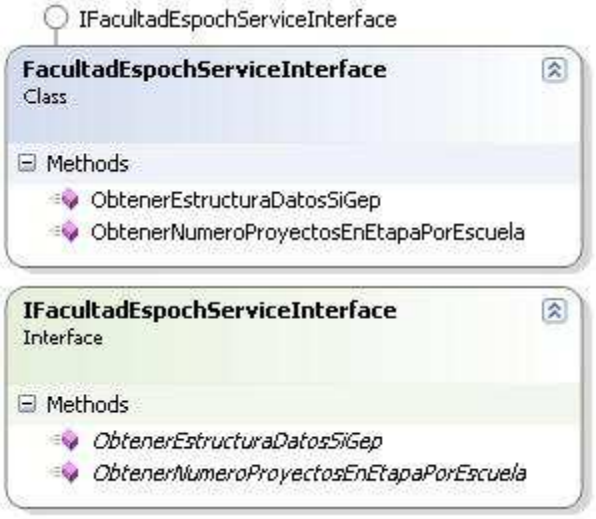


Figura V.36.-"**Interface de servicio:** IFacultadEspochServiceInterface"

Figura V.37.-"**Interface de servicio:** IMatriculaServiceInterface"

Figura V.38.-"**Interface de servicio:** IMiembroServiceInterface"

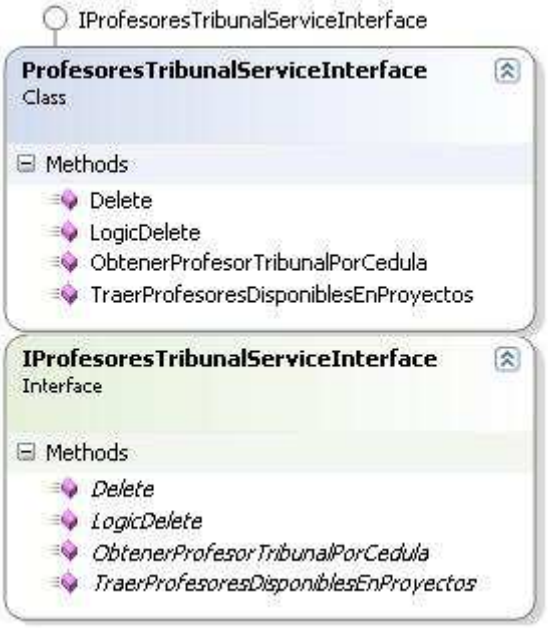


Figura V.39.-"Interface de servicio: IProfesoresTribunalServiceInterface**"**

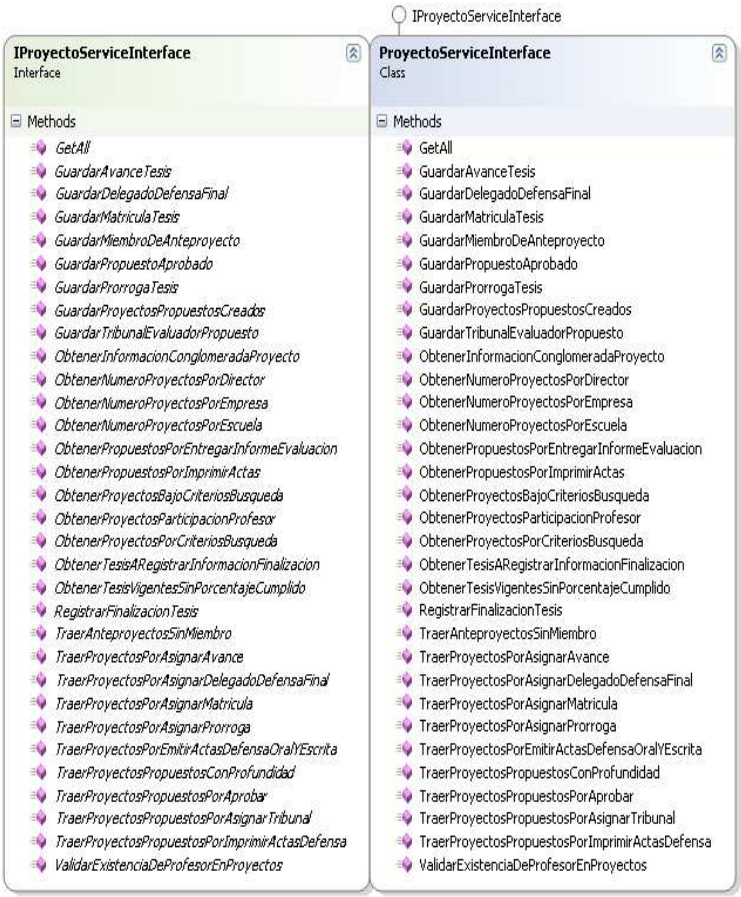
Figura V.40.-"Interface de servicio: IRolProfesorServiceInterface"

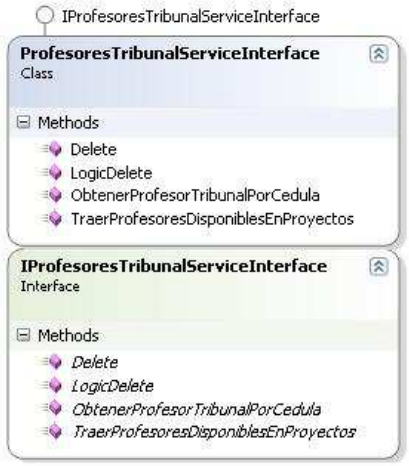
5.4 ITERACION

A continuación se detalla las iteraciones realizadas para el desarrollo de los servicios de nuestro sistema.

5.5.1 PRIMERA ITERACION

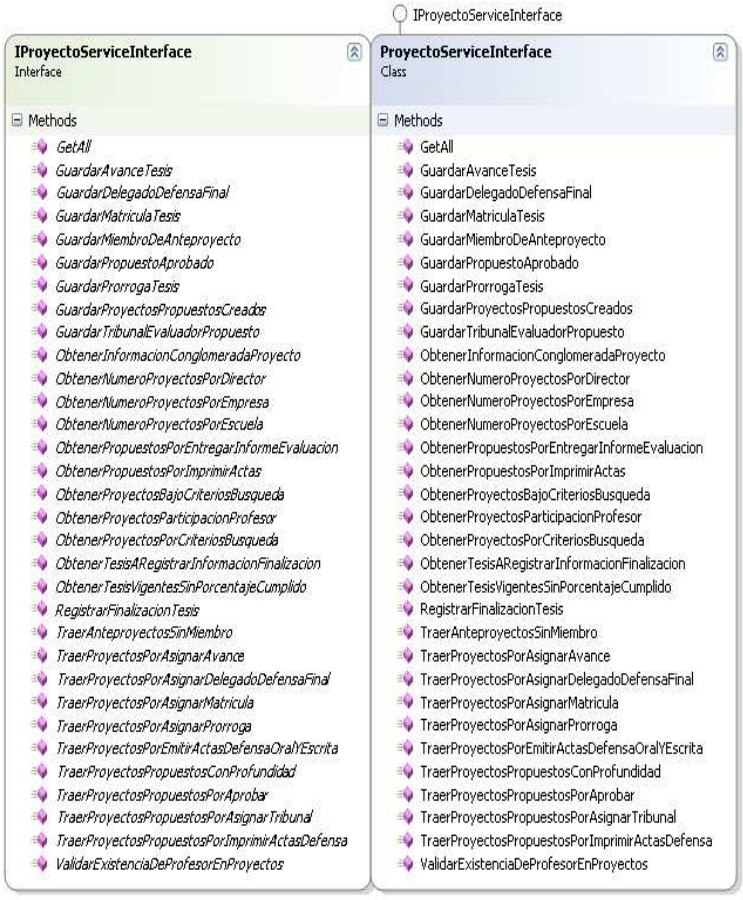
Tabla V.33. – "Resultado de servicios: 1 iteración".

Descripción	Servicios Publicados
Gestiona información relacionada al Proyecto cuando este se encuentra en Etapa de Propuesto, posee métodos para GuardarProyectosPropuestosCreados, GuardarPropuestoAprobado, GuardarTribunalEvaluadorPropuesto, GuardarMiembroAnteproyecto.	
Servicio que gestiona información relacionada al Miembro o Asesor, posee métodos de Consulta.	

Servicio que gestiona información relacionada a los Profesores del Tribunal evaluador de un Proyecto Propuesto, posee métodos de Consulta.	

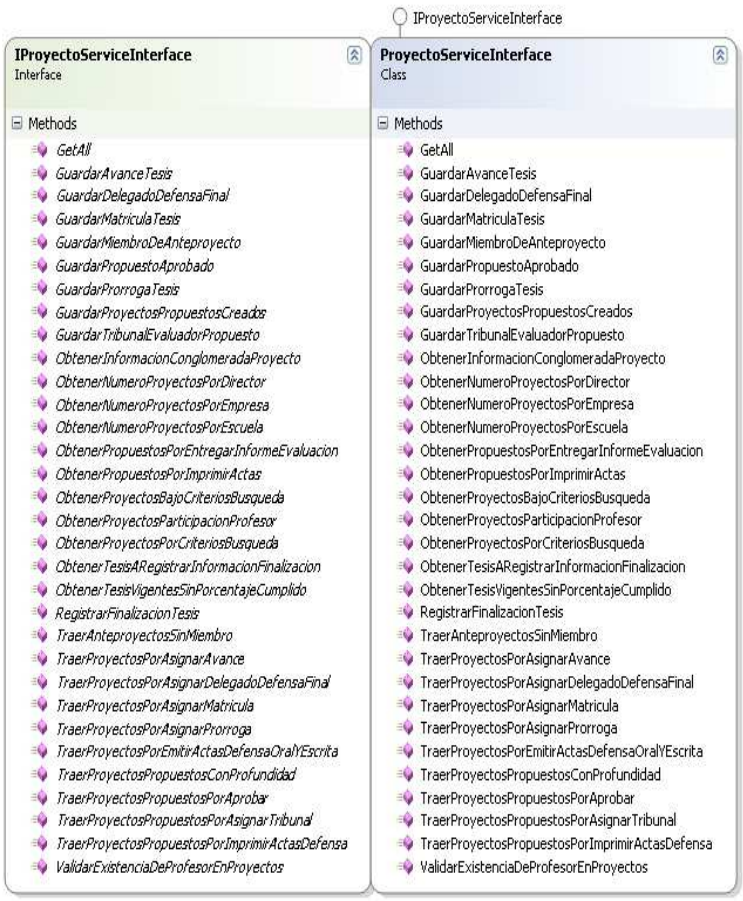
5.5.2 SEGUNDA ITERACION

Tabla V.34. – "Resultado de servicios : 2 iteración".

Descripción	Servicios Publicados
Gestiona información relacionada a Matriculas de un Proyecto, posee metodos para TraerProyectosPorAsignarMatricula GuardarMatriculaTesis	

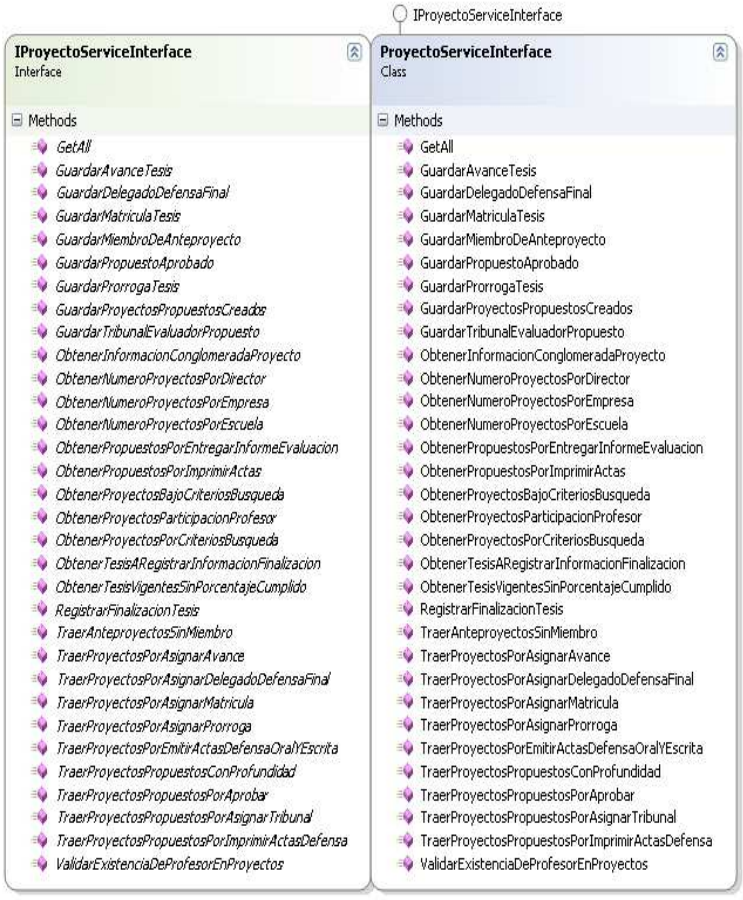
5.5.3 TERCERA ITERACION

Tabla V.35. – "Resultado de servicios : 3 iteración".

Descripción	Servicios Publicados
Gestiona información relacionada a Matriculas de un Proyecto, posee metodos para TraerProyectosPorAsignarAvance GuardarAvanceTesis	

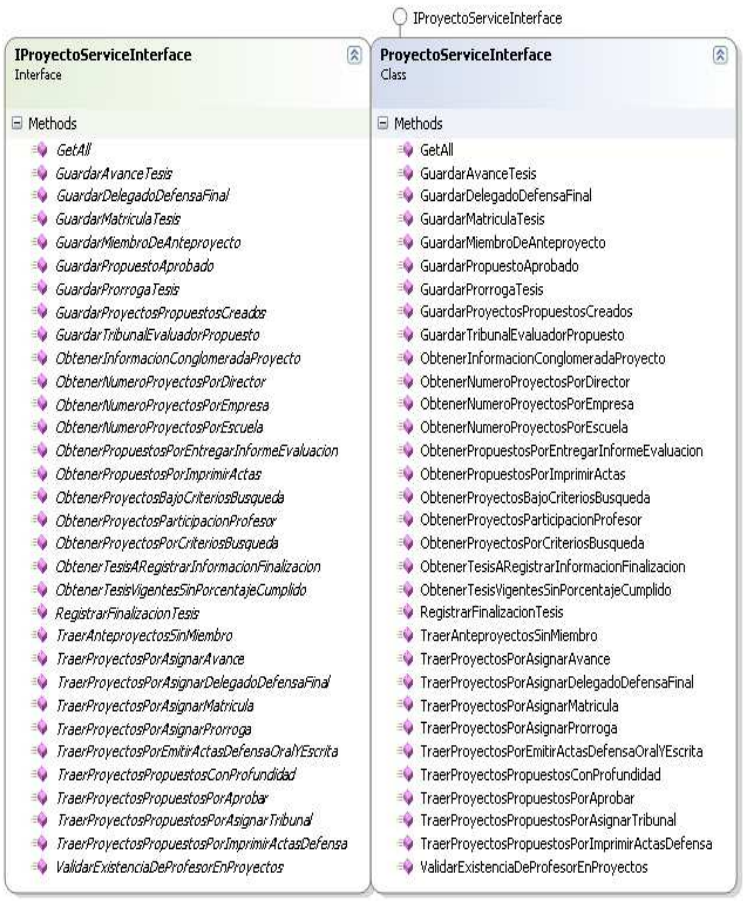
5.5.4. CUARTA ITERACION

Tabla V.36. – "Resultado de servicios: 4 iteración".

Descripción	Servicios Publicados
Proporciona métodos para cumplir tareas de finalización de tesis descritos a continuacion ObtenerTesisARegistrarInformacionFinalizacion RegistrarFinalizacionTesis	

5.5.5. QUINTA ITERACION

Tabla V.37. – "Resultado de servicios : 5 iteración".

Descripción	Servicios Publicados
Proporciona métodos de busqueda de Proyectos factible de impresión de Actas ObtenerPropuestosPorImprimirActas TraerProyectosPorEmitirActas DefensaOralYescrita	

5.5.6. SEXTA ITERACION

Tabla V.38. – "Resultado de servicios : 6 iteración".

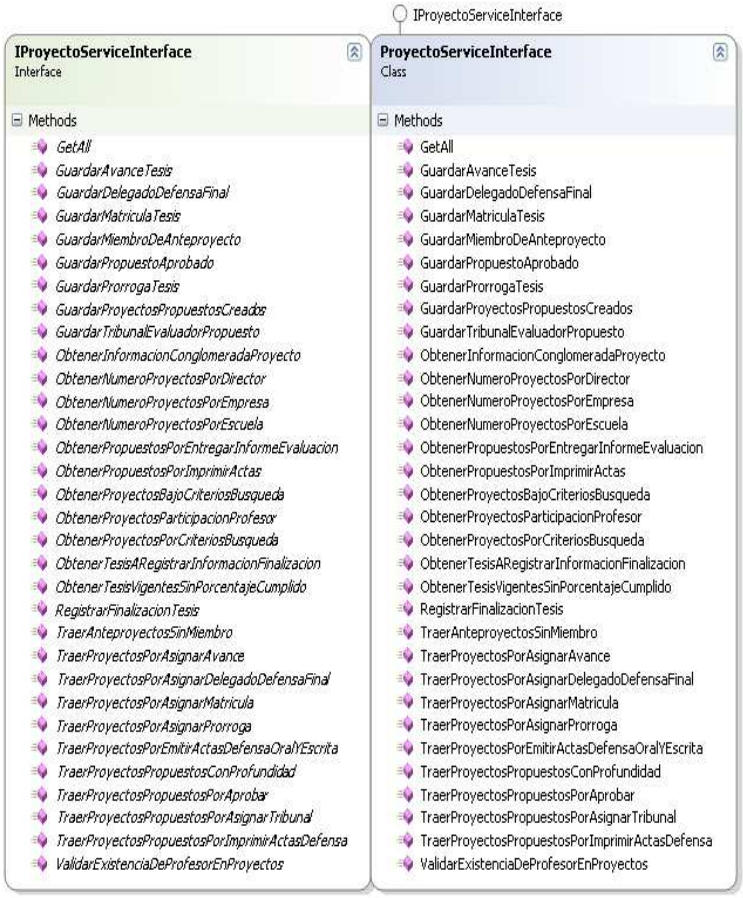
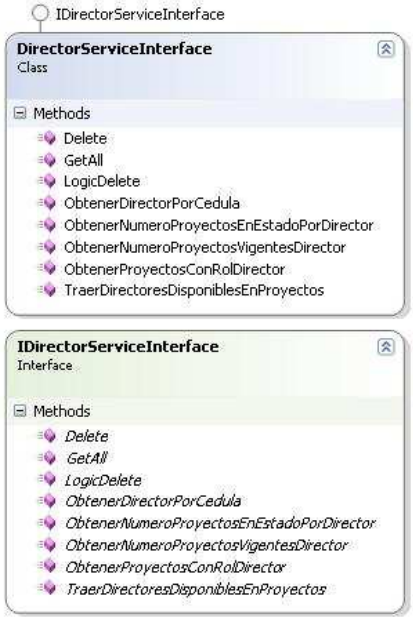
Descripción	Servicios Publicados
Proporciona métodos de búsqueda de Proyectos factible de impresión de Actas ObtenerNumeroProyectosPorDirector ObtenerNumeroProyectosPorEmpresa ObtenerNumeroProyectosPorEscuela	
Posee métodos para consultar información de Director, asi como de Delegados.	

Tabla V.39. – "Resultado de servicios : 6 iteración".

Descripción	Servicios Publicados
Proporciona métodos para administrar información de Empresas que serán utilizadas en la creación de Proyectos Propuestos.	

ANÁLISIS DE RESULTADOS

Un problema a la hora de construir una aplicación orientadada a servicios es comprobar si la aplicación construida realmente es una aplicación SOA. Para comprobar si una aplicación lo es, la mejor forma de hacerlo es verificando que la aplicación cumpla con los Principios de la Orientación a Servicios**.**

Los Principios de la orientación a servicios según Thomas Erl son:

- **Los Servicios deben ser reusables:**

Todo servicio debe ser diseñado y construido pensando en su reutilización dentro de la misma aplicación, dentro del dominio de aplicaciones de la empresa o incluso dentro del dominio público para su uso masivo.

Los servicios de nuestro aplicativo SIGEP fueron diseñados y construidos bajo la premisa de la reutilización por ejemplo en la figura V.4.1 podemos observar como el servicio: MiembroServiceInterface puede ser accedido y reutilizado por otros

aplicativos.

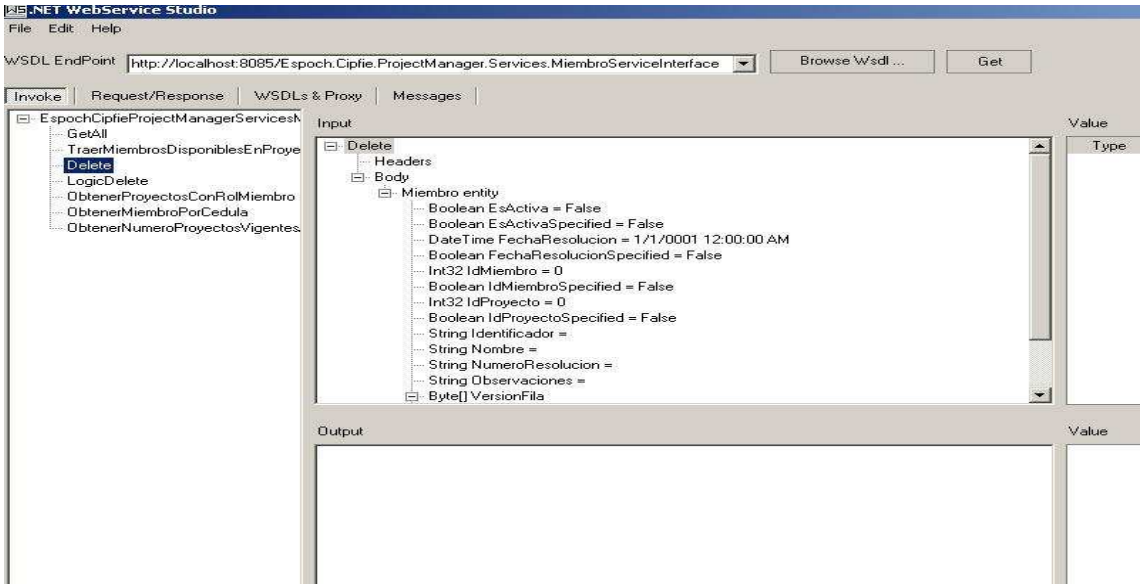


Figura V.41.- "Reutilización del servicio: **MiembroServiceInterface**"

- **Los Servicios deben proporcionar un contrato formal:**

Todo servicio desarrollado, debe proporcionar un contrato en el cual figuren: El nombre del servicio, su forma de acceso, las funcionalidades que ofrece, los datos de entrada de cada una de las funcionalidades y los datos de salida. De esta manera, todo consumidor del servicio, accederá a este mediante el contrato, logrando así la indepencia entre el consumidor y la implementación del propio servicio.

Los servicios de nuestro aplicativo SIGEP exponen un contrato o wsdl, en la figura V.42 observamos el contrato del servicio *MiembroServiceInterface.*

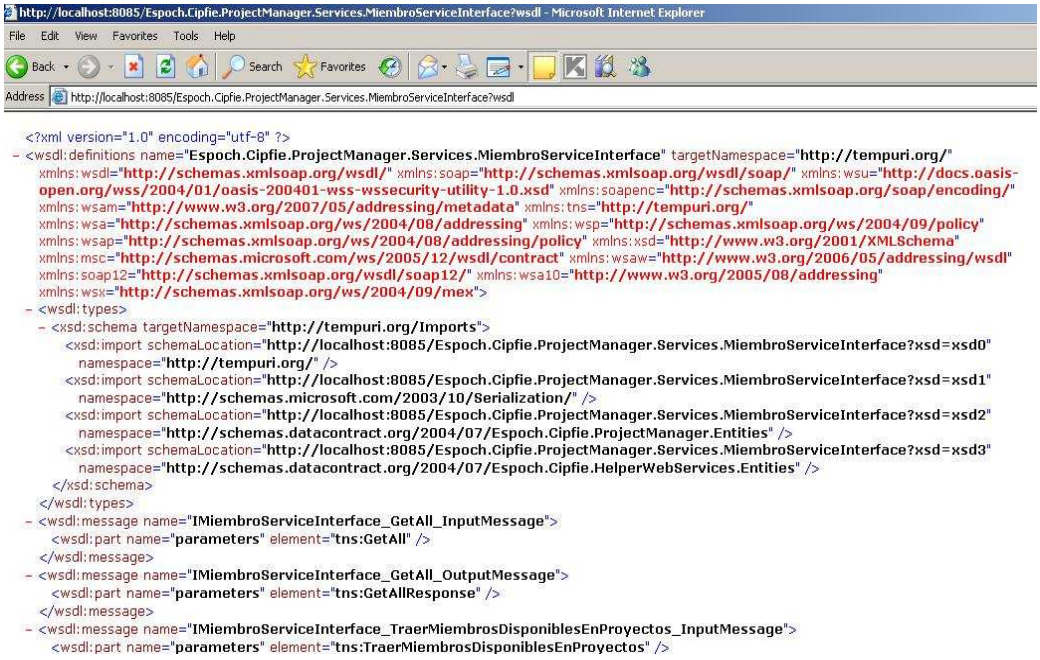


Figura V.42.- "Contrato del servicio: **MiembroServiceInterface**"

- **Los Servicios deben tener bajo acoplamiento:**

  Los servicios tienen que ser independientes los unos de los otros. Para lograr ese bajo acoplamiento, lo que se hará es que cada vez que se vaya a ejecutar un servicio, se accederá a él a través del contrato, logrando así la independencia entre el servicio que se va a ejecutar y el que lo llama. Si conseguimos este bajo acoplamiento, entonces los servicios podrán ser totalmente reutilizables.

  La figura V.43 y la figura V.44 muestran los servicios ProyectoServiceInterface y EstudianteServiceInterface respectivamente, cada servicio expone un contrato único e independiente en nuestro aplicativo SIGEP**.**

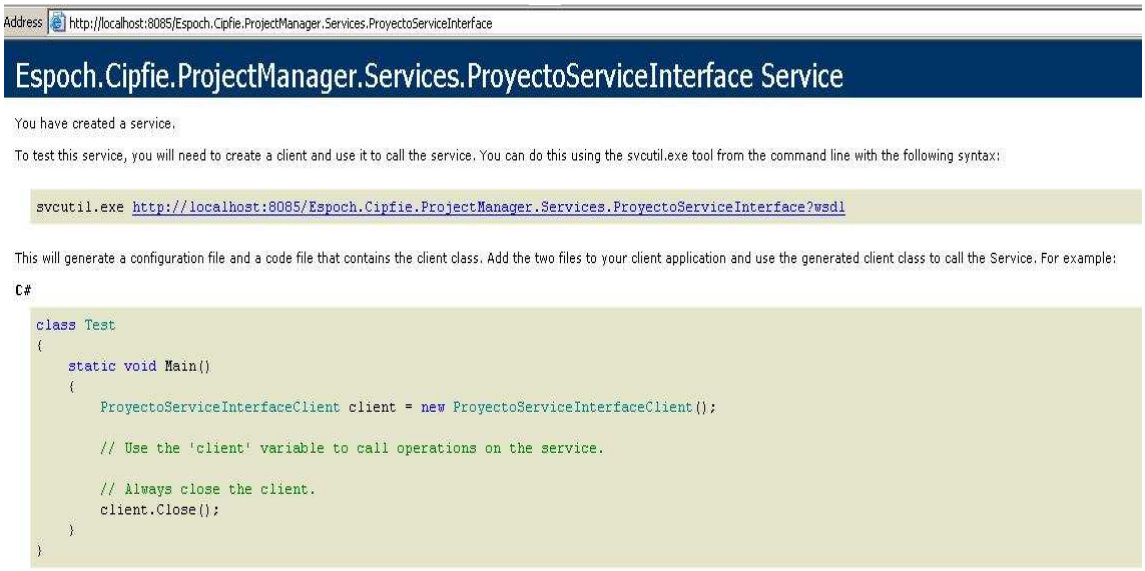


Figura V.43.- "ProyectoServiceInterface"

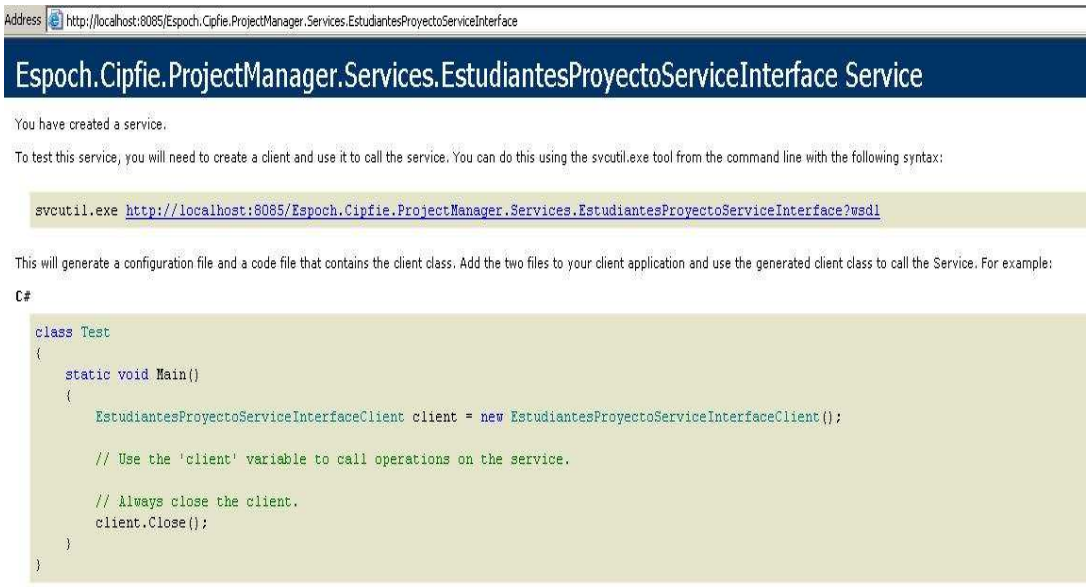


Figura V.44.- "EstudianteProyectoServiceInterface"

- **Los Servicios deben de ser autónomos:**

  Todo Servicio debe tener su propio entorno de ejecución. De esta manera el servicio es totalmente independiente y nos podemos asegurar que así podrá ser reutilizable desde el punto de vista de la plataforma de ejecución.

  La figura V.45 muestra el host o entorno de ejecución de nuestros servicios en el aplicativo SIGEP**.**

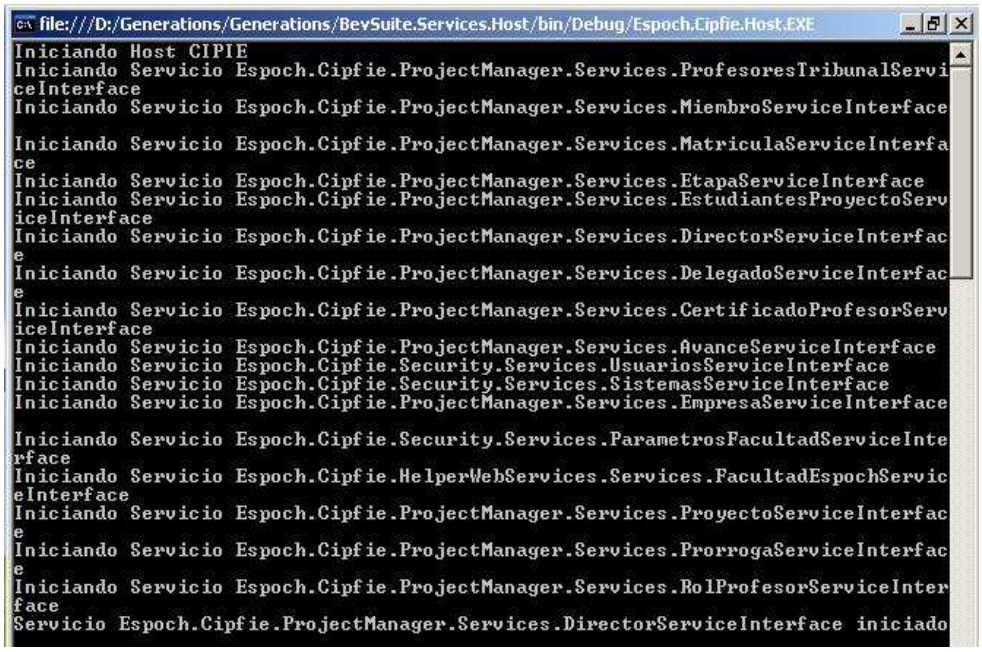


Figura V.45.- "Host o entorno de ejecución de los servicios del aplicativo SIGEP"

- **Los Servicios deben permitir la composición:**

  Todo servicio debe ser construido de tal manera que pueda ser utilizado para construir servicios genéricos de más alto nivel, el cual estará compuesto de servicios de más bajo nivel.

  Los servicios del aplicativo SIGEP al tener bajo acoplamiento y ser autónomos permiten la composición.

- **Los Servicios no deben tener estado:**

  Un servicio no debe guardar ningún tipo de información. Esto es así porque una aplicación está formada por un conjunto de servicios, lo que implica que si un servicio almacena algún tipo de información, se pueden producir problemas de inconsistencia de datos. La solución, es que un servicio sólo contenga lógica, y que toda información esté almacenada en algún sistema de información sea del tipo que sea.

La figura V.46 muestra la distribución arquitectónica de nuestro aplicativo SIGEP donde podemos observar las diferentes capas del aplicativo SIGEP notando que tenemos una capa única para los servicios.

Figura V.46.- "Distribución arquitectónica del aplicativo SIGEP"

- **Los Servicios deben poder ser descubiertos:**

  Todo servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades. En el caso de los Servicios Web, el descubrimiento se logrará publicando los interfaces de los servicios en registros UDDI.

  Los servicios del aplicativo SIGEP, al ser servicios web están en la capacidad de ser publicados en los registros UDDI de la ESPOCH.

Cuando se desarrollan aplicaciones SOA es muy útil y necesario tener en cuenta siempre estos principios, ya que nos van a dar las pautas necesarias para tomar ciertas decisiones de diseño complejas.

Tabla V.40. – "Resultado de índices SOA del aplicativo SIGEP".

INDICES SOA APLICADO AL SIGEP		
INDICES	**% CUMPLIMIENTO SIGEP**	**% TOTAL**
• Los Servicios deben ser reusables	**100 %**	**14.28 %**
• Los Servicios deben proporcionar	**100 %**	**14.28 %**

un contrato formal		
• Los Servicios deben tener bajo acoplamiento	**100 %**	**14.28 %**
• Los Servicios deben permitir la composición	**100 %**	**14.28 %**
• Los Servicios deben de ser autónomos:	**100 %**	**14.28 %**
• Los Servicios no deben tener estado:	**100 %**	**14.28 %**
• Los Servicios deben poder ser descubiertos:	**100 %**	**14.28 %**
TOTAL:		**100 %**

Según los datos obtenidos en la tabla **Tabla V.40** podemos observar que la aplicación desarrollada **SIGEP** cumple con los principios de la orientación a servicios.

CONCLUSIONES

- El interés de las organizaciones ya no está limitado únicamente al desarrollo de software que automatice determinadas actividades individuales, sino que por el contrario, tienen como objetivo final la automatización de todo el proceso de negocio, ya que de ello depende en gran parte su competitividad.

- El desarrollo de software orientado a servicios esta formado por un conjunto de procesos de negocio. A su vez estos procesos de negocio están compuestos por servicios que proporcionan las operaciones que se necesitan ejecutar para que el proceso de negocio llegue a buen término. Para ejecutar esas operaciones es necesario el envío de los datos necesarios mediante los correspondientes mensajes.

- La orientación a servicios, significa tres cosas. En primer lugar, el desarrollo de aplicaciones puede realizarse más rápidamente. En segundo lugar, la integración se hace más fácil. Y finalmente, la reutilización de aplicaciones y procesos es una realidad.

- Windows Communication Foundation es un modelo para crear aplicaciones orientadas a servicios que permite a los desarrolladores que actualmente crean aplicaciones orientadas a objetos crear también aplicaciones orientadas a servicios de una manera similar. WCF conduce a los Servicios Web a un nivel superior, proporcionando a los desarrolladores una alta productividad para construir aplicaciones seguras, fiables e interoperables, simplificando sustancialmente el proceso de creación de sistemas distribuidos.

- La programación extrema es una metodología cuya filosofía es satisfacer al completo las necesidades del cliente, menos orientados al documento, exigiendo una cantidad más pequeña de documentación. De muchas maneras son más bien orientados al código: siguiendo un camino que dice que la parte importante de la documentación es el código fuente.

- La metodología propuesta nos ha permitido integrar los conceptos de orientación a servicios implementados con el modelo de programación unificada de Windows Communication Foundation, los principios y mejores practicas de la metodología ágil XP. Resultado de la cual obtenemos el sistema orientado a servicios Sigep que constituye un valor importante dentro del CIPFIE.

- Según los datos obtenidos en la tabla Tabla V.40. podemos observar que la aplicación desarrollada SIGEP permite cumplir un 100% los principios de la orientación a servicios por tanto la metodología propuesta cumple con el objetivo para la creación de software orientado a servicios.

RECOMENDACIONES

- Analizar el grado de mejoramiento y aceptación de las ventajas que ofrece la arquitectura orientada a servicios dentro del desarrollo de aplicaciones empresariales, con el fin de comenzar una implementación adecuada del sistema a realizarse.

- Se debe efectuar un estudio minucioso sobre las tecnologías que permiten desarrollar aplicaciones empresariales existentes en el mercado, para de esa manera obtener la más indicada en el momento de desarrollar el sistema y así cumplir las necesidades existentes.

- Realizar pruebas piloto encaminadas a determinar problemas antes de emprender cualquier proyecto real. De igual manera evaluar los riesgos que el proyecto puede correr al realizar este tipo de soluciones.

- Sería recomendable hacer un estudio similar al presente con la nueva tecnología que propone IBM, utilizando así la herramienta *WebSphere Business Modeler* para el análisis y diseño de los procesos de negocio; *WebSphere Integration Developer* para el desarrollo de los servicios.

RESUMEN

Diseñar una metodología orientado a servicios que permita el desarrollo de software aplicado a la construcción del Sistema de Gestión de Proyectos (SIGEP), es el objeto de esta investigación.

Se utilizó metodología Ágil XP, la tecnología Windows Communication Foundation (WCF) y los principios de la orientación a servicios. Se procedió a adaptar el paradigma de la orientación a servicios para implementar la tecnología WCF, aplicando el método científico.

Se llegó a diseñar una metodología basada en servicios Web, que integra a los servicios y procesos de negocio en una sola fase llamada Modelado del Negocio, que al ser implementada en gestión de anteproyectos, proyectos e información de tesis del Centro de Investigación y Producción de la Facultad de Informática y electrónica (CIPFIE) en la Escuela Superior Politécnica del Chimborazo (ESPOCH), permite automatizar de manera eficaz y eficiente procedimientos manuales utilizando la reusabilidad, bajo acoplamiento, composición y autonomía del servicio en un 100% integrable y reutilizable por cualquier otro sistema y permitiendo reducir en un 40% el tiempo y los recursos gracias a su implantación.

Esta metodologia puede adoptarse como guía para desarrolladores de software garantizando la construcción independientemente de la tecnología que apliquen

SUMARY

The objective of this designing a methodology oriented to services permitting the software development applied to the constructions of Project Management System (SIGEP). The methodology Agil XP, the technology Windows Communication Foundation (WCF) and the service-oriented principles were used. The service orientation paradigm was adapted to implement the WCF technology applying the scientific method. A methodology based on web services which involves the business services and processes into only one phase called Business Modeling was designed. Upon implementing it in the project, preliminary sketch management and thesis information of the investigation and Production Center of the Informatics and Electronics Faculty (CIPFIE) in the Escuela Superior Politecnica de Chimborazo (ESPOCH) permits to automate efficiently and effectively handy procedures using re-use, under service matching, composition and autonomy by 100% through

In conclusion, this methodology can be adopted as a guide for developers to ensure the development of software-oriented services regardless of the technology applied.

We recommend using this method in conjunction with other technologies for the development of services.

GLOSARIO DE TERMINOS

Assembly. Net

En el framework Microsoft de .NET una assembly es una biblioteca de código parcialmente compilada para el uso en despliegue. Un assembly es un PE (portátil ejecutable) considerado en el archivo. Hay dos tipos, assembly del proceso (EXE) y assembly de biblioteca (DLL).

Back – end.

El Back - end es la parte que procesa la entrada desde el front-end. La conexión del front-end y el back-end es un tipo de interfaz.

Capa.

Una capa se puede concebir como un patrón de arquitectura en el que los componentes utilizan servicios en las capas inferiores. La utilización de capas facilita el mantenimiento. La comunicación entre dos capas determina la facilidad con que se podrá particionar la aplicación en ese punto para la distribución física a través de los niveles. Unos esquemas de capas estrictos no permiten a las capas tener acceso a otras capas que no sean las inmediatamente inferiores, mientras que unos esquemas de capas más flexibles permiten a una capa determinada utilizar cualquier otra que esté por debajo de ella.

Código Administrado.

Propiedades personalizadas que señalan a un ensamblado de código administrado

Compilado - Ensamblado.

Es el proceso por el cual se traduce un conjunto de instrucciones llamado código fuente a código objeto. Normalmente se utiliza el termino compilado para referirse a la tarea de compilación en conjunto al proceso de enlazado, ya que la mayoría (por no decir todos) de los compiladores realizan por defecto ambas tareas en conjunto, al menos que se especifique que se desea únicamente obtener el código objeto.

Componente.

Dicho de forma sencilla, un componente es una parte de un sistema. Una definición más específica de componente es una unidad de funcionalidad que se puede amortizar a través de diversas implementaciones. Un componente generalmente se implementa como un objeto de software que expone una o más interfaces y que implementa la lógica.

Common Language Runtime (CLR).

El componente más importante dentro del .NET Framework es algo llamado el Common Language Runtime (CLR). Si se está familiarizado con JAVA se puede pensar en términos de que el CLR es el equivalente el la plataforma .NET del JAVA Virtual Machine (JVM).

CRUD.

CRUD responde a las siglas en inglés de Crear, Leer, Actualizar y Eliminar. Se refiere a las operaciones que se pueden realizar en un almacén de datos. En la terminología de SQL, Crear, Leer, Actualizar y Eliminar se refieren a INSERTAR, SELECCIONAR, ACTUALIZAR y ELIMINAR operaciones, respectivamente.

Enrutamiento dinámico de datos (DDR).

El enrutamiento dinámico de datos es la lógica que se utiliza para determinar a qué servidor de base de datos se enviará una recuperación de datos o una solicitud de modificación cuando los datos se encuentran particionados entre diversos servidores. El DDR se puede implementar mediante la utilización de un algoritmo hash, una tabla de reglas o algún otro esquema de partición.

Ensamblado.

Un ensamblado es una unidad de implementación en una aplicación basada en .NET framework.

Flujo de trabajo.

El flujo de trabajo se refiere a un proceso empresarial en el que los pasos se deben realizar en un determinado orden, y se deben cumplir unas condiciones predefinidas, antes de avanzar de un paso al siguiente.

Front – end.

El Front-end es la parte del software que interactúa con el usuario. La conexión del front-end es un tipo de interfaz.

Ingeniería de software.

Es la aplicación práctica del conocimiento científico en el análisis, diseño y construcción de aplicaciones informáticas y la documentación necesaria requerida para desarrollar, operar y mantener.

Interfaz de Usuario.

Una interfaz de Usuario es un punto de interacción del usuario y el sistema.

Interoperable.

Capaz de funcionar en diferentes sistemas servidor y cliente

Mensaje.

Un mensaje es una unidad de información que se transmite electrónicamente de un servicio a otro.

.NET Framework.

El Framework de .NET da la estructura básica sobre la cual la visión de Microsoft está establecida. Asumiendo que la visión de .NET se haga realidad, en un futuro cercano el mundo entero estará dependiente del Internet, existirán accesos de banda ancha desde cualquier parte del mundo.

Objeto.

Un objeto es una representación detallada, concreta y particular de un "algo". Tal representación determina su identidad, su estado y su comportamiento particular en un momento dado.

La identidad de un objeto le permite ser distinguido de entre otros y esto se da gracias al nombre que cada objeto posee.

El estado de un objeto es el conjunto de valores concretos que lo caracterizan en un momento dado, como peso, color, precio, etc.

Sistema.

Es el conjunto de elementos relacionados entre sí funcionalmente, de modo que cada elemento del sistema es función de algún otro elemento, no habiendo ningún elemento aislado. Se debe considerar un sistema de computación e información como el conjunto de componentes físicos (hardware), lógicos (software), de

comunicación (redes de cualquier tipo o Internet) y medios humanos, todo ello unido permite el tratamiento de información.

ANEXOS

Apéndice A – Especificación de Requisitos software

ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

SISTEMA GESTIÓN DE PROYECTOS

SIGEP

DOCUMENTO DE ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

Autores	*Juan Pablo Tixe A, Luis Alberto Rivera R.*
Fecha inicial	*Martes, 15 de Abril del 2008*
Versión:	*1.0*
Archivo:	*ESPOCH-SRS-SiGep.doc*

Control de Cambios

Registro de Cambios

Fecha (dd/mm/aaaa)	**Responsable**	**Versión**	**Sección**	**Descripción de los Cambios**
15/04/2008	Juan Pablo Tixe A. Luis Alberto Rivera R.	1.0	Definiciones, Restricciones, Suposiciones	Documento Inicial

Registro de Revisiones

Nombre	**Versión Revisada**	**Conforme**	**Cargo**	**Fecha**

Distribución

Nombre	**Cargo**

Aprobado por

Nombre	**Cargo**	**Fecha** (dd/mm/aaaa)	**Firma**

Tabla de Contenido

1. Introducción

1.1. Propósito

El presente documento definirá de manera clara los requerimientos funcionales y generales así como sus restricciones para el Sistema SiGep.

1.2. Alcance

SiGep pretende automatizar el proceso correspondiente a la gestión de Tesis pertenecientes a la Facultad de Informática y Electrónica; así mismo estará integrado al Sistema Académico de la Escuela Superior Politécnica de Chimborazo con el fin de utilizar información relacionada a Docentes y Estudiantes.

1.3. Definiciones, Acrónimos y abreviaturas

1.3.1. Definiciones

Sistema Académico Espoch	Sistema Informático que se encarga de gestionar la información referente a Docentes, Estudiantes; misma que será utilizada por SiGep.

Tabla 1.1 Definiciones.

1.3.2. Acrónimos y abreviaturas

ESPOCH	Escuela Superior Politécnica de Chimborazo
CIPFIE	Centro de Investigación y Producción de la Facultad de Informática y Electrónica.
SIGEP	Sistema de Gestión de Proyectos.

Tabla 1.2 Acrónimos y abreviaturas.

1.4. *Referencias*

El estándar de este documento se basa en las siguientes especificaciones:
IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE std. 830, 1998.

2. Descripción General

En esta sección se presenta una descripción general del sistema donde se muestran las principales áreas de negocio a las cuales el sistema debe dar soporte, las funciones que el sistema debe realizar, la información utilizada, las restricciones y otros factores que afectan al desarrollo del mismo.

2.1. *Perspectiva del Sistema*

Una vez puesto en producción el SiGep, se espera administrar de mejor manera la información correspondiente a Proyectos de Tesis propuestos por Estudiantes de la Facultad de Informática y Electrónica.

2.2. *Funciones del Sistema*

2.2.1 Anteproyectos de Tesis

Permitirá realizar acciones como el registro de un Proyecto Propuesto, Asignación de Tribunal Evaluador, etc.; sobre Proyectos que se encuentren en esta Etapa.

2.2.2. Proyectos de Tesis

Permitirá realizar acciones relacionadas al registro de Matricula, Avances; así como la Asignación de Delegados para Defensa Final y el registro de su finalización de Proyectos.

2.2.3. Documentación

Facilitará la impresión de documentos necesarios para el flujo del proceso relacionado a los Proyectos de Tesis.

2.2.4. Reportes

Presenta reportes estadísticos a nivel gerencial con el fin de apoyar en la toma de decisiones.

2.2.5. Consultas

Proporciona el acceso a consultas generales de Proyectos de Tesis; presentando información dinámica en dependencia de su etapa.

2.2.6. Mantenimiento

Permite administrar los parámetros y catálogos del Sistema.

2.3. *Perfiles de usuario*

A continuación se describe brevemente los usuarios que interactuaran con el Sistema a implementarse:

2.3.1 Administrador

Perfil de usuario encargado de administrar y parametrizar el SiGep.

2.3.1 Supervisor

Perfil de usuario al cual se le concederá permisos a Reportes Estadísticos y Consultas relacionadas a Proyectos de Tesis.

2.3.1 Secretaria

Perfil de usuario al cual se le concederá permisos sobre las acciones correspondientes a la gestión de proyectos en todas sus Etapas.

2.4. *Restricciones*

- SiGep no contempla reglas de Negocio basadas de otros Centros de Investigación de la ESPOCH.

2.5. *Suposiciones y dependencias*

Las Reglas de Negocio del SiGep están basadas en el Reglamento Actual del CIPFIE.

3. Requerimientos específicos

3.1. *Interfaces externas*

3.1.1. Interfaz de usuarios

Para la integración con el Sistema Académico de la ESPOCH se utilizará Servicios Web proporcionados por dicha Institución.

3.1.2. Interfaz de hardware de producción

Clientes
Procesador Intel Pentium III
Memoria 256Mb
5Gb disco duro

Servidores de Aplicaciones
Procesador Intel Pentium IV de 32 bits
Memoria 2Gb
Disco duro 30 Gb

Servidores de base de datos
Procesador Intel Pentium IV de 32 bits
Memoria 2 Gb
Disco duro 150 Gb

3.1.3. Interfaces de Software producción

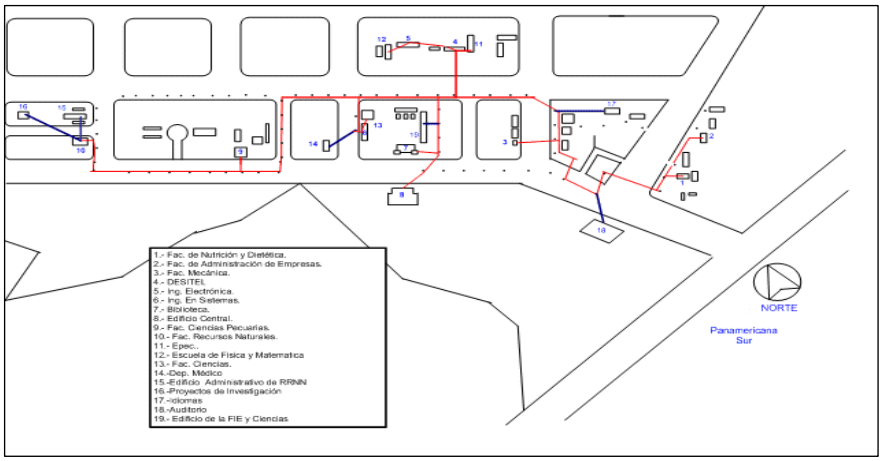
Servidores de base de datos
Sistema Operativo Windows
Motor de base de datos SQL Server 2000

Servidores de lógica de negocio
Sistema operativo Windows Server 2003 service pack 1
Internet Information Services
Microsoft .Net Framework 3.0

Computadores cliente
Sistema operativo Windows XP Service pack 2
Microsorft .Net Framework 3.0

3.1.4. Interfaz de comunicación

La ESPOCH cuenta con una red que se presenta en el siguiente gráfico:



3.2 *Requerimientos Funcionales*

A continuación se especifica el detalle del alcance de cada una de las opciones del Sistema, se definen datos de entrada, procesos y salidas.
Campos Requeridos (R).
Campos Opcionales (O).

3.2.1. REQ 1: Anteproyectos de Tesis

3.2.1.1. REQ 1.1: *Crear Proyecto Propuesto*

3.2.1.1.1. Entradas

- Nombre del Proyecto. (R)
- Escuela. (R)
- Carrera. (R)
- Área. (R)
- Lugar de Realización. (R)
- Director del Proyecto. (R)
- Estudiantes proponentes (R)

3.2.1.1.2. Procesos

- Verificar campos requeridos.
- Validar la no existencia de un proyecto con el mismo nombre.
- Enviar a guardar los datos.

3.2.1.1.3. Salidas

- Mensaje de resultado de proceso.

3.2.1.2. REQ 1.2: *Asignar Tribunal Evaluador*

3.2.1.2.1. Entradas

- Proyecto Seleccionado (R)
- Fecha de Evaluación. (R)
- Fecha de Entrega de Actas. (R)
- Docentes pertenecientes al Tribunal. (R)

3.2.1.2.2. Procesos

- Validar fecha entrega de Actas no menor a la fecha Evaluación.
- Validar la existencia de Docentes en Tribunal asignado.
- Enviar a guardar datos.

3.2.1.2.3. Salidas

- Mensaje de resultado de proceso.

3.2.1.3. REQ 1.3: *Registrar Informe Evaluación*

3.2.1.3.1. Entradas

- Proyecto Seleccionado. (R)
- Fecha de Evaluación. (R)
- Calificación. (R)
- Observaciones. (O)
- Nuevo Tema. (O)

3.2.1.3.2. Procesos

- Verificar existencia de datos requeridos.
- Validar fecha de Evaluación no menor a fecha de asignación.
- Enviar a guardar los datos.

3.2.1.3.3. Salidas

- Mensaje de resultado de proceso.

3.2.1.4. REQ 1.4: *Registrar Asesor Anteproyecto*

3.2.1.4.1. Entradas

- Proyecto Seleccionado. (R)

- Número Resolución. (R)
- Fecha Resolución. (R)
- Identificador Asesor. (O)
- Nombre Asesor. (O)

3.2.1.4.2. Procesos

- Verificar existencia de datos requeridos.
- Validar la no participación del Asesor asignado en el proyecto con otros roles.
- Enviar a guardar los datos.

3.2.1.4.3. Salidas

- Mensaje de resultado de proceso.

3.2.2. REQ 2: Proyectos de Tesis

3.2.2.1. REQ 2.1: *Matricular Proyecto*

3.2.2.1.1. Entradas

- Proyecto seleccionado. (R)
- Número Comprobante. (R)
- Fecha Resolución. (R)
- Fecha Fin Matricula. (R)
- Observaciones. (O)

3.2.2.1.2. Procesos

- Verificar tipo de datos correctos.
- Validar fecha fin no menor a fecha resolución.
- Enviar a guardar datos.

3.2.2.1.3. Salidas

- Mensaje de resultado de proceso.

3.2.2.2. REQ 2.2: *Registrar Avance*

3.2.2.2.1. Entradas

- Proyecto seleccionado. (R)
- Porcentaje. (R)
- Fecha Avance. (R)

- Observaciones. (O)

3.2.2.2.2. Procesos

- Verificar existencia de campos requeridos.
- Validar suma de porcentaje no mayor a 100.
- Validar fecha de avance no menor a fecha de avance anterior.
- Enviar a guardar datos.

3.2.2.2.3. Salidas

- Mensaje de resultado de proceso.

3.2.2.3. REQ 2.3: *Asignar Delegado Defensa*

3.2.2.3.1. Entradas

- Proyecto seleccionado. (R)
- Fecha de Defensa asignado. (R)
- Fecha Entrega Actas calificación. (R)
- Número Resolución. (R)
- Fecha Resolución. (R)
- Identificador Delegado. (R)
- Nombre Delegado. (R)

3.2.2.3.2. Procesos

- Verificar existencia de campos requeridos.
- Validar consistencia en fechas asignadas.
- Enviar a guardar datos.

3.2.2.3.3. Salidas

- Mensaje de resultado de proceso.

3.2.2.4. REQ 2.4: *Registrar Información Final*

3.2.2.4.1. Entradas

- Proyecto seleccionado. (R)
- Código de Tesis Finalizada. (R)
- Fecha Finalización. (R)
- Resumen. (R)

- Palabras Claves. (R)
- Calificación Proyecto. (R)

3.2.2.4.2. Procesos

- Verificar existencia de campos requeridos.
- Validar fecha finalización no mayor a la actual.
- Validar calificación no mayor a 20.

3.2.2.4.3. Salidas

- Mensaje de resultado de proceso.

3.2.3. REQ 3: Documentos

3.2.3.1. REQ 3.1: *Acta Evaluación Propuesto*

3.2.3.1.1. Entradas

- Proyecto seleccionado. (R)

3.2.3.1.2. Procesos

- Buscar información relacionada al proyecto seleccionado.
- Cargar datos del Proyecto en respectivo Reporte.

3.2.3.1.3. Salidas

- Reporte cargado.

3.2.3.2. REQ 3.2: *Acta Evaluación Escrita Tesis*

3.2.3.2.1. Entradas

- Proyecto Seleccionado. (R)

3.2.3.2.2. Procesos

- Buscar información relacionada al proyecto seleccionado.
- Cargar datos del Proyecto en respectivo Reporte.

3.2.3.2.3. Salidas

- Reporte cargado.

3.2.3.3. REQ 3.3: *Acta Evaluación Oral*

3.2.3.3.1. Entradas

- Proyecto seleccionado. (R)

3.2.3.3.2. Procesos

- Buscar información relacionada al proyecto seleccionado.
- Cargar datos del Proyecto en respectivo Reporte.

3.2.3.3.3. Salidas

- Reporte cargado.

3.2.3.4. REQ 3.4: *Certificado Tesis Dirigidas por Docente*

3.2.3.4.1. Entradas

- Identificador del Docente. (R)

3.2.3.4.2. Procesos

- Buscar tesis finalizadas pertenecientes al docente.
- Imprimir certificado.

3.2.3.4.3. Salidas

- Certificado impreso.

3.2.4. REQ 4: Consultas

3.2.4.1. REQ 4.1: Listado Propuestos Sin Evaluación

3.2.4.1.1. Entradas

3.2.4.1.2. Procesos

- Buscar Proyectos Propuestos que han sido asignados un Tribunal Evaluador y aun no han sido registrado su factibilidad.
- Presentar listado de Proyectos encontrados.

3.2.4.1.3. Salidas

- Listado de Proyectos encontrados .

3.2.4.2. REQ 4.2: *Listado de Tesis Vigentes*

3.2.4.2.1. Entradas

3.2.4.2.2. Procesos

- Buscar Proyectos en Etapa Tesis que aun no han sido finalizadas.
- Presentar listado de Proyectos encontrados.

3.2.4.2.3. Salidas

- Listado de Proyectos encontrados.

3.2.4.3. REQ 4.3: *Búsqueda General Proyectos*

3.2.4.3.1. Entradas

- Etapa. (O)
- Estado Tesis. (O)
- Nombre Proyecto. (O)
- Identificación Estudiante. (O)
- Nombre Estudiante. (O)
- Lugar Realización. (O)

3.2.4.3.2. Procesos

- Buscar Proyectos que coincidan con los criterios de búsqueda ingresados.
- Presentar listado de Proyectos encontrados.

3.2.4.3.3. Salidas

- Listado de Proyectos encontrados.

3.2.5. REQ 5: Reportes

3.2.5.1. REQ 5.1: Proyectos Por Empresas

3.2.5.1.1. Entradas

- Identificador de Empresa. (R)
- Fecha Inicio. (R)

- Fecha Fin. (R)

3.2.5.1.2. Procesos

- Buscar Proyectos pertenecientes a la Empresa.
- Generar Reporte Estadístico con la información encontrada.
- Presentar Reporte Estadístico.

3.2.5.1.3. Salidas

- Reporte Estadístico.

3.2.5.2. REQ 5.2: *Proyectos Por Escuelas*

3.2.5.2.1. Entradas

- Identificador de la Escuela. (R)
- Fecha Inicio. (R)
- Fecha Fin. (R)

3.2.5.2.2. Procesos

- Buscar proyectos pertenecientes a la Escuela.
- Generar Reporte Estadístico con la información encontrada.
- Presentar Reporte Estadístico.

3.2.5.2.3. Salidas

- Reporte Estadístico.

3.2.5.2. REQ 5.2: *Proyectos Por Director*

3.2.5.2.1. Entradas

- Identificador del Docente. (R)
- Fecha Inicio. (R)
- Fecha Fin. (R)

3.2.5.2.2. Procesos

- Buscar proyectos pertenecientes al Docente.
- Generar Reporte Estadístico con la información encontrada.

- Presentar Reporte Estadístico.

3.2.5.2.3. Salidas

- Reporte Estadístico.

3.2.6. REQ 6: Mantenimiento

3.2.6.1. REQ 6.1: *Mantenimiento Empresas*

3.2.6.1.1. Entradas

- Siglas. (R)
- Nombre. (R)
- Persona Contacto. (R)
- Teléfono Contacto. (R)
- Ubicación Referencia. (R)

3.2.6.1.2. Procesos

- Verificar existencia de información requerida.
- Verificar la no existencia de siglas repetidas.
- Verificar la no existencia de nombres de Empresa repetidos.
- Enviar a guardar información recabada.

3.2.6.1.3. Salidas

- Mensaje de resultado de proceso.

BIBLIOGRAFIA

LIBROS

1. ZAPATA, J. Integración de Aplicaciones desarrolladas sobre tecnologías y plataformas de software heterogéneas con el uso de XML Web Services. Caso Practico: ESPOCH:(Tesis) (Ing. en Sistm Inf). Escuela Superior Politécnica de Chimborazo. Facultad de Informática y Electrónica. 2004. pp. 339 - 343.

2. VALLE, L. Estudio de la arquitectura Orientada A Servicios, Aplicado En la Escuela Superior Militar "Eloy Alfaro" (Tesis) (Ing. En Sistm. Inf). Escuela Superior Politécnica de Chimborazo. Facultad de Informática y Electrónica. 2006, pp. 270.

3. BETTETINI, G. y COLOMBO, G. Las Nuevas Tecnologías de la comunicación. Paidós-Barcelona. s.f. pp. 100-220.

4. SKIBO, Craig. Et al. Working with Microsoft Visual Studio 2005. USA. Microsoft Press. 2006. 304p

5. LOWY J. Programming .NET Components. USA: O' Reilly, 2005. 480 p.

6. MICROSOFT CORPORATION, STOECKER. M. MCAD / MCSD Self-Paced Training Kit: Developing Windows-Based Applications with Microsoft Visual Basic.NET and Microsoft Visual C#.NET. 02. ed. USA: Microsoft Press, 2003. 834 p.

7. NOYES B. Data Binding with Windows Forms 2.0: Programming Smart Client Data Applications with .NET. USA: Addison Wesley Professional, 2006. 736 p.

8. PRESSMAN, R., Ingeniería del Software un Enfoque Práctico. España: McGraw-Hill, 1993. pp. 49 – 58.

PUBLICACIONES EN EL WORLD WIDE WEB

9. **Arquitectura de aplicaciones de .NET: Diseño de aplicaciones y servicios. MSDN Microsoft.**
http://www.microsoft.com/spanish/msdn/arquitectura/das/guias/AppArchCh1.asp
http://www.microsoft.com/spanish/msdn/arquitectura/das/guias/AppArchCh2.asp
http://www.microsoft.com/spanish/msdn/arquitectura/das/guias/AppArchCh3.asp
http://www.microsoft.com/spanish/msdn/arquitectura/das/guias/AppArchCh4.asp
http://www.microsoft.com/spanish/msdn/arquitectura/das/guias/AppArchCh5.asp
(2004).

10. **Arquitectura - Abobe System**
http://www.adobe.com/es/enterprise/xml.html
(2006).

11. **Document Solution- Conexión entre las personas, la información y los procesos**
http://www.alabsa.com/solutions/solutions_document.php
(2005)

12. **Estilos y Patrones en la Estrategia de Arquitectura de Microsoft**
http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp
(2005).

13. **Introducción a la Arquitectura de Software**

http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp

(2006).

14. **XML**

http://www.bibliodgsca.unam.mx/tesis/tes7cllg/sec_26.htm