Universidad Central “Marta Abreu” de Las Villas
Facultad Matemática, Física y Computación
Ingeniería en Informática

TRABAJO DE DIPLOMA

Modelado y Diseño de Servicios en una Arquitectura Orientada a Servicios para un Sistema de Control de la Formación de Másteres y Especialistas de Postgrado.

AUTOR: Osmel Mora Cuellar

TUTOR: Dr. Rosendo Moreno Rodríguez

Santa Clara
2010
“Año 52 de la Revolución”

Dictamen

Hago constar que el presente trabajo fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de los estudios de la especialidad de Ingeniería Informática, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Tutor Firma del Jefe del Laboratorio

Dedicatoria

"A mi madre por tener fe en mi".

Agradecimientos

Agradezco a mis familiares,
por su apoyo y en especial a mis padres.
A mi tutor por brindarme su ayuda incondicional.
A mi novia y a su familia que sin ellos no hubiera sido posible.
A mis amigos y todas las personas que me ayudaron a seguir adelante.

Resumen

Con este trabajo se pretende revelar y diseñar una serie de servicios que deben ser implementados al desarrollar un Sistema de Control de la Actividad de Postgrado bajo los preceptos de la Arquitectura Orientada a los Servicios (SOA). Específicamente se toma como objeto de estudio el proceso sustantivo “Formación de Másteres y Especialistas”, por ser el más avanzado en la determinación de sus procedimientos de forma automatizada, lo que favorece la determinación de esos servicios, que de hecho son comunes para los otros procesos sustantivos de Postgrado: “Formación Doctoral” y “Superación Profesional”, y pueden ser reutilizados en otros sistemas de información que se acometan para la Universidad.

Palabras Claves

Ingeniería del Software, Arquitectura Orientada a los Servicios (SOA), Servicios Web.

Abstract

This work is sought to reveal and to design a series of services that should be implemented when developing a System for Control of Postgraduate Activity under the precepts from the Services Oriented Architecture (SOA). Specifically takes like study object the process "Formation of Masters and Specialists", to be the most advanced in the determination of their procedures in an automated way, what favours the determination of those services that in fact are common for the other Postgraduate processes: "Doctoral formation" and "Professional Study", and they can be reuse in other information systems developed for the University.

Key words

Software Engineering, Service Oriented Architecture (SOA), Web Service

Tabla de Contenidos

Contenido	Página
Introducción	1
CAPÍTULO 1. VISIÓN GENERAL DE LA ARQUITECUTA ORIENTADA A LOS SERVICIOS.	6
1.1 Evolución de los sistemas distribuidos.	6
1.1.1 *Sockets*	6
1.1.2 *Remote Procedure Call* (RPC)	7
1.1.3 *Distributed Component Object Model* (DCOM)	8
1.1.4 *Common Object Request Broker Architecture* (CORBA)	9
1.1.5 *Java/RMI* y *Java IDL*	11
1.1.6 .*Net Remoting*	13
1.1.7 *XML-RPC*	14
1.1.8 *Web Services*	15
1.2 Arquitectura Orientada a Servicios (SOA).	18
1.2.1 Evolución de SOA.	18
1.2.2 Principios de la orientación a servicios.	22
1.2.3 Beneficios de SOA.	22
1.2.4 Cómo se resuelven los retos de SOA.	24
1.2.5 Capas de servicios.	24
1.2.6 Estrategias de desarrollo.	27
1.3 SOA y la gestión de procesos de negocios (BPM).	30
1.4. Conclusiones Parciales	31
CAPÍTULO 2. ANÁLISIS ORIENTADO A SERVICIOS DEL PROCESO DE MAESTRÍAS Y ESPECIALIDADES.	32
2.1 Análisis orientado a servicios.	32
2.2. Conclusiones Parciales	56
CAPÍTULO 3: DISEÑO ORIENTADO A SERVICIOS DEL SISTEMA DE GESTIÓN DE POSTGRADO.	57
3.1 Diseño de servicios de negocio centrados en entidades.	58

3.2 Diseño de servicios de aplicación. 65
3.3 Diseño de servicios de negocio centrados en tareas. 73
3.4. Conclusiones Parciales 79
CONCLUSIONES 80
RECOMENDACIONES 81
REFERENCIAS BIBLIOGRÁFICAS 82
BIBLIOGRAFÍA 83
ANEXOS -

Lista de Figuras

1. Figura 1.1. Tabla descriptora de ficheros de un proceso.
2. Figura 1.2. Mecanismo de Llamada a Procedimiento Remoto.
3. Figura 1.3. Aplicación distribuida RMI.
4. Figura 1.4. Arquitectura *.NET Remoting.*
5. Figura 1.5. Arquitectura XML-RPC.
6. Figura 1.6. Descubrimiento e invocación de servicios Web.
7. Figura 1.7. Un antiguo modelo de SOA.
8. Figura 1.8. Las tres capas primarias de servicios.
9. Figura 1.9. Estrategia de desarrollo de arriba hacia abajo.
10. Figura 1.10. Estrategia de desarrollo de abajo hacia arriba.
11. Figura 1.11. Estrategia de desarrollo ágil.
12. Figura 2.1. Proceso de análisis orientado a servicios.
13. Figura 2.2. Procedimiento de gestación y aprobación de nuevos programas de Maestrías y Especialidades.
14. Figura 2.3. Sub-procedimiento de planificación de inicio de Ediciones de Programas de Maestrías y Especialidades.
15. Figura 2.4. Sub-procedimiento oficial de inicio de Ediciones de Programas de Maestrías y Especialidades.
16. Figura 2.5. Procedimiento de ejecución y control en las áreas de Maestrías y Especialidades.
17. Figura 2.6. Pasos a seguir al modelar los servicios candidatos.
18. Figura 2.7. Servicios de negocio candidatos.
19. Figura 2.8. Servicio candidato Procesamiento de Documento.
20. Figura 2.9. Servicio candidato Actualización.
21. Figura 2.10. Escenario de configuración.
22. Figura 2.11. Composición del Proceso de Maestrías y Especialidades.
23. Figura 3.1. Servicios de negocio centrados en entidades.
24. Figura 3.2. Tipo de dato complejo PPGA.
25. Figura 3.3. WSDL del servicio Programa de Postgrado Académico.
26. Figura 3.4. Tipo de dato complejo Plantillas_Ediciones.

27. Figura 3.5. WSDL del servicio Planillas de Ediciones.
28. Figura 3.6. Tipo de dato complejo Exp_Estudiante.
29. Figura 3.7. Tipo de dato complejo ExpToUpdate.
30. Figura 3.8. WSDL del servicio Expediente de Estudiantes.
31. Figura 3.9. Servicios de aplicación.
32. Figura 3.10. Tipo de dato complejo FileToUpdate.
33. Figura 3.11. Tipos de datos complejos Estudiante_Baja y ListaActualizada.
34. Figura 3.12. WSDL del servicio Actualización.
35. Figura 3.13. Tipos de datos complejos Request_info y Response_info.
36. Figura 3.14. WSDL del servicio Buscar.
37. Figura 3.15. TrasformToXMLType y TransformToXMLReturnCodeType.
38. Figura 3.16. WSDL del servicio Buscar.
39. Figura 3.17. Servicios de negocio centrados en tareas.
40. Figura 3.18. Relación entre Gestación y Aprobación con Programa de Postgrado Académico y Procesamiento de Documento.
41. Figura 3.19. Diagrama de secuencia.
42. Figura 3.20. Peticiones y respuestas identificadas en el servicio Gestación y Aprobación.
43. Figura 3.21. Interfaz del servicio Gestación y Aprobación.
44. Figura 3.22. Relación del servicio Inicio y Matrícula con Plantillas de Ediciones, Acta de Exámenes, Lista de Matrícula, Expediente de Estudiante y Expediente de Ediciones.
45. Figura 3.23. Diagrama de secuencia.
46. Figura 3.24. Peticiones y respuestas identificadas en el servicio Inicio y Matrícula.
47. Figura 3.25. Interfaz del servicio Inicio y Matrícula.
48. Figura 3.26. Relación del servicio *Ejecución y Control* con *Sábana de Notas* y S*ábana de Créditos*.
49. Figura 3.27. Diagrama de secuencia.
50. Figura 3.28. Peticiones y respuestas identificadas en el servicio Ejecución y Control.
51. Figura 3.29. Interfaz del servicio Ejecución y Control.

INTRODUCCION

Planteamiento del Problema:

Hoy en día las empresas necesitan poder interconectar los procesos, personas e información tanto con la propia organización como atravesando sus fronteras. La falta de integración entre los componentes de Tecnologías de Información (IT, *Information Technologies*) –sistemas, aplicaciones y datos- hace difícil obtener una respuesta rápida y efectiva ante los cambios que afectan de forma natural a los negocios. La inflexibilidad genera costes, reduce la capacidad de respuesta ante los clientes, compromete el cumplimiento con las normativas legales y afectan negativamente a la productividad de los empleados.

En suma, una deficiente integración es uno de los problemas más importantes a los que las organizaciones deben hacer frente para mantener su competitividad y garantizar su crecimiento. La Arquitectura Orientada a Servicios (SOA, *Service Oriented Architecture*) supone una estrategia general de organización de los elementos de IT, de forma que una colección abigarrada de sistemas distribuidos y aplicaciones complejas se pueda transformar en una red de recursos integrados, simplificada y sumamente flexible. Un proyecto SOA bien ejecutado permite alinear los recursos de IT de forma más directa con los objetivos de negocio, ganando así un mayor grado de integración con clientes y proveedores, proporcionando una inteligencia de negocio más precisa y más accesible con la cual se podrán adoptar mejores decisiones, y ayuda a las empresas a optimizar sus procesos internos y sus flujos de información para mejorar la productividad individual. El resultado neto es un aumento muy notable de la agilidad de la organización.(Microsoft 2006)

La Arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante servicios Web -lo que no quiere decir que cuando se implementan servicios Web se está en presencia de SOA, aunque se potencian mutuamente-, una tecnología basada en estándares

e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas en un conjunto de servicios que se implementan en forma modular.

¿Qué es un servicio exactamente? Un servicio es una funcionalidad concreta que puede ser descubierta en la red y que describe tanto lo que puede hacer como el modo de interactuar con ella.(Fensel, Lausen et al. 2007)

La estrategia de orientación a servicios permite la creación de servicios y aplicaciones compuestas que pueden existir con independencia de las tecnologías subyacentes. En lugar de exigir que todos los datos y lógica de negocio residan en un mismo ordenador, el modelo de servicios facilita el acceso y consumo de los recursos de IT a través de la red. Puesto que los servicios están diseñados para ser independientes, autónomos y para interconectarse adecuadamente, pueden combinarse y recombinarse con suma facilidad en aplicaciones complejas que respondan a las necesidades de cada momento en el seno de una organización. El resultado final es que las organizaciones que adoptan la orientación a servicios pueden crear y reutilizar servicios y aplicaciones y adaptarlos ante los cambios evolutivos que se producen dentro y fuera de ellas, y con ello adquirir la agilidad necesaria para ganar ventaja competitiva.(Box, Vadoss et al. 2007)

SOA y la Gestión de Procesos de Negocio (BPM, *Business Process Management*) son complementarios y permiten optimizar las aportaciones de cada uno gracias a sus propias virtudes. En efecto, SOA es una aproximación dirigida por las IT mientras que el BPM se orienta hacia el negocio, la unión de estos dos conceptos permite entonces una mejor adecuación entre los objetivos de los servicios informáticos y los objetivos de negocio. Además, la comunicación entre los dos está entonces asegurada con el objetivo común de mejorar el rendimiento.(Erl 2005)

Una de las actividades más complejas actualmente en la educación Superior, lo es sin duda la Educación de Postgrado, el conocido cuarto nivel de educación. En sus inicios, el MES estableció un reglamento que establecía las figuras de Cursos, Entrenamientos y estudios de Postgrado, aunque también desde la década del 80 se establecieron reglamentaciones adicionales para los Doctorados, en todos los casos de forma tutelar y muy similar a la forma soviética de grados científicos. Pero en décadas posteriores surgieron nuevas figuras como los Diplomados, las Maestrías y las Especialidades y

actualmente se suman los Doctorados Curriculares y los Postdoctorales. En muchas Universidades del país y en específico en esta UCLV la matrícula de postgrado supera ya la matrícula de pregrado, por lo que la gestión de esta actividad es compleja, engorrosa, y requiere de mucha seguridad y confiabilidad.

Esta necesidad motiva que se haga necesario un Sistema Automatizado para la gestión de esta actividad, el cual ha sido diseñado e implementado con las más modernas técnicas de computación, y probado durante más de 20 años, después de su implantación, incluyendo su perfeccionamiento y mantenimiento continuo, en el Centro de estudios de Informática primero y en la Facultad de Matemática, Física y Computación de la UCLV, posteriormente.

Este sistema tuvo tres versiones anteriores que fueron desarrolladas en DataEase 4.53 para DOS, FoxPro 2.6 para Windows, y Access'97 para Windows. La primera versión cumplió con los objetivos de los años 80 pero en la actualidad dicho sistema no es utilizado, no solo por estar desarrollado sobre DOS, sino porque en su concepción se tuvo en cuenta el primer reglamento de Postgrado que no incluía toda una serie de actividades actuales (Maestrías, Doctorados, entre otros) y los informes de estadística actuales; sin embargo fue utilizado para la emisión automatizada del libro de ofertas de postgrado que en aquellos tiempos se distribuía entre todas las empresas y organismos. El segundo fue un sistema que también sirvió de preámbulo y su instalación fue parcial. Ya la tercera versión fue totalmente explotada desde un primer comienzo con Access'97, hasta su perfeccionamiento con la versión de Access'2000, que incluía la resolución de postgrado del año 1996. Posterior a esto se le hicieron varios perfeccionamientos para Access XP, y 2003, que mejoraron las estadísticas finales, implementaron varios informes oficiales (Certificados de Postgrados, Certificaciones de Estudios Concluidos de Maestrías, etc.).(Moreno 2006)

El MES durante mucho tiempo trabajó de forma paralela en el desarrollo de otros sistemas de gestión de la información docente y administrativa, pero recientemente ha creado un grupo de desarrollo de un Sistema para la Gestión de la Información de la Nueva Universidad (SIGENU). Otras Universidades han desarrollado otros módulos, pero en el caso de la UCLV fue encargado de la creación de un nuevo Sistema de Control del

Postgrado sobre tecnologías de Software Libre con ambiente Web. Ese trabajo es un encargo del MES para el Departamento de Desarrollo de Sistemas (DPS) de la UCLV.

La Dirección de Postgrado de la UCLV ha trabajado de conjunto con la DPS para la creación de ese nuevo sistema, y por ello desarrollo el análisis y diseño basado en UML del mismo, pero además de forma paralela ha trabajado en la organización del proceso de gestión de toda la información de esta actividad y como resultado de esto, en el curso pasado se trabajó en la definición de una Ontología de Dominio y Modelado de Procesos para el Negocio de Postgrado en la UCLV, y en este se trabajó además en la Modelación de los Procesos de Negocios (BPM) para el proceso sustantivo de Formación de Másteres y Especialistas. Otra arista de enfrentar este problema es la determinación de los servicios que debe realizar un sistema como este, para que el desarrollo del mismo sea enfrentado desde la óptica de la Arquitectura Orientada a los Servicios (SOA).

Objetivo General:

Analizar y diseñar los servicios asociados a un sistema automatizado para la gestión del proceso sustantivo de formación de másteres y especialistas, de manera de acometer su desarrollo en base a la arquitectura orientada a servicios.

Objetivos Específicos:

1. Analizar las actividades relacionadas con los procedimientos del Proceso Sustantivo de Formación de Másteres y Especialidades.
2. Modelar los servicios candidatos y las capas de abstracción que respondan a la lógica del proceso de negocio.
3. Diseñar las definiciones WSDL de los servicios de la arquitectura SOA propuesta en la herramienta XMLSpy 2009 sp1.

Preguntas de Investigación:

1. ¿Podrán establecerse con claridad las actividades relacionadas con los procedimientos del Proceso Sustantivo de Formación de Másteres y Especialistas?
2. ¿Podrán abstraerse los servicios candidatos para un sistema de gestión de la información de esta actividad en base a SOA?

3. ¿Se podrán diseñar las definiciones WSDL de los servicios de la arquitectura SOA propuesta en la herramienta XMLSpy 2009 sp1?

Justificación de la Investigación:

El enfoque de desarrollo basado en SOA es novedoso en nuestro medio actualmente y nunca abordado para el desarrollo de un sistema de gestión de la actividad de postgrado, por tanto es importante e interesante llevar a cabo esta investigación en aras de promover la aplicación de esta tecnología en otras esferas posteriormente.

Viabilidad de la Investigación:

Se tiene información primaria esencial sobre la Arquitectura Orientada a los Servicios y sobre el problema en cuestión objeto de estudio que servirá de base para acometer la investigación.

Hipótesis de Investigación:

La definición de los servicios candidatos y el diseño de las definiciones WSDL de estos servicios de la arquitectura SOA propuesta en la herramienta XMLSpy 2009 sp1 a ser usados en un sistema de control del Proceso Sustantivo de Formación de Másteres y Especialistas de Postgrado, ayudará a obtener un sistema mucho más cohesivo en sus aspectos de control e independiente de ciertas tareas que pueden servir de apoyo a otros sistemas de gestión universitaria.

Organización de la tesis

Para un mejor ordenamiento de las ideas y conclusiones de este trabajo se determinó organizarlo en tres capítulos que incluyen lo siguiente:

1. Capítulo 1: Dedicado al estudio y síntesis del Marco Teórico relacionado con la temática de la Arquitectura Orientada a los Servicios (SOA).
2. Capítulo 2: Dedicado al análisis orientado a servicios de los procedimientos del Proceso de Formación de Másteres y Especialistas.
3. Capítulo 3: Dedicado al diseño orientado a servicio de los procedimientos del Proceso de Formación de Másteres y Especialistas.

CAPÍTULO 1. VISIÓN GENERAL DE LA ARQUITECUTA ORIENTADA A LOS SERVICIOS.

CAPÍTULO 1. VISIÓN GENERAL DE LA ARQUITECUTA ORIENTADA A LOS SERVICIOS.

1.1 Evolución de los sistemas distribuidos.

La Arquitectura Orientada a Servicios (SOA) no se trata de un software, una plataforma o un lenguaje de programación, SOA es un tipo de arquitectura tecnológica que cumple con los principios de la orientación a servicios. Cuando se implementa a través de la plataforma tecnológica *Web Services*, SOA brinda el potencial para soportar y proveer estos principios a lo largo de los procesos de negocio y los dominios de automatización de una empresa.(Erl 2005)

A continuación se realiza un análisis de las tecnologías de intercambio de datos entre aplicaciones distribuidas, con el objetivo de comprender la evolución de los sistemas distribuidos, sus aportes y carencias hasta llegar a la actualidad, donde los servicios *Web* son la base tecnológica de las arquitecturas orientadas a servicios.

1.1.1 Sockets

Los *sockets* (enchufe) aparecieron a principios de los 80 con el sistema UNIX de Berkeley, para proporcionar un medio de comunicación entre los procesos. Los *sockets*, si hacemos un símil, tienen la misma función que la que pudiera tener la comunicación por correo o por teléfono. El *socket*, por tanto, ofrece dos puntos de contacto entre distintas aplicaciones, a través de los cuales estas se comunican.

Otra característica importante de los puntos de comunicación concierne al conjunto de otros puntos que permite acceder. La noción de dominio de un *socket* permite definir el conjunto de *sockets* con los cuales se podrá establecer una comunicación por medio de él.

Un *socket* es un punto de comunicación por el cual un proceso puede emitir o recibir información. En el interior de un proceso, un *socket* se identifica por un descriptor de la misma naturaleza que los que identifican los archivos, al igual que todos los procesos del sistema UNIX de Berkeley.(Hall 2001)

La comunicación mediante *sockets* es una interfaz (o servicio) con la capa de transporte (nivel 4) de la jerarquía OSI. La filosofía de la división por capas de un sistema es

encapsular, dentro de cada una de ellas, detalles que conciernen sólo a cada capa, y presentársela al usuario de tal forma que este pueda trabajar con ella sin necesidad de conocer sus detalles de implementación. La interfaz de acceso a la capa de transporte del sistema UNIX de Berkeley no está totalmente aislada de las capas inferiores, por lo que a la hora de trabajar con *sockets*, es necesario conocer algunos detalles sobre esas capas. En concreto, a la hora de establecer una conexión mediante *sockets*, es necesario conocer la familia o dominio de la conexión, y el tipo de conexión.

La creación de un *socket* se realizará por la primitiva *socket*, cuyo valor de vuelta es un descriptor sobre el cual es posible realizar operaciones de escritura y lectura. Un *socket* permite la comunicación en los dos sentidos (conexión *full-dúplex*).

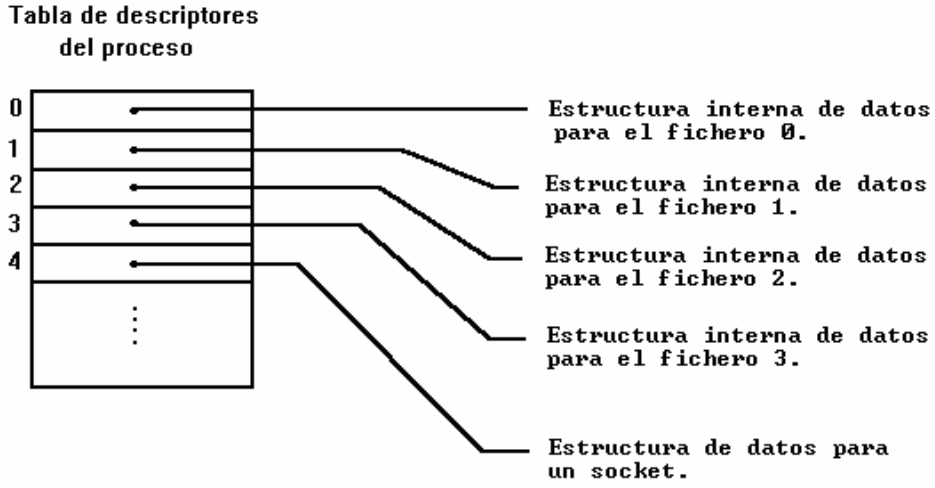

Figura 1.1. Tabla descriptora de ficheros de un proceso.

1.1.2 Remote Procedure Call (RPC).

El **RPC** (*Remote Procedure Call*, **Llamada a Procedimiento Remoto**) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los *sockets* usados hasta el momento de su creación. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.

Las RPC son muy utilizadas dentro del paradigma cliente-servidor. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando éste de vuelta el resultado de dicha operación al cliente.

Un RPC es análogo a una llamada de función. Como una llamada de función, cuando un RPC es hecho, los argumentos de la llamada son pasados al procedimiento remoto y la persona que llama espera la respuesta devuelta por el procedimiento remoto. La figura 1.2 muestra el flujo de actividad que tiene lugar durante una llamada RPC entre dos sistemas en red. El cliente hace una llamada de procedimiento que envía una petición para el servidor y espera. El hilo es bloqueado desde el procesamiento hasta que se reciba una respuesta, o se acabe el tiempo. Cuando la petición llega, el servidor llama una rutina de despacho que realiza el servicio pedido, y envía la respuesta al cliente. Después de que la llamada RPC es completada, el programa del cliente continúa. RPC específicamente da soporte a las aplicaciones de la red.(Marshal 1999)

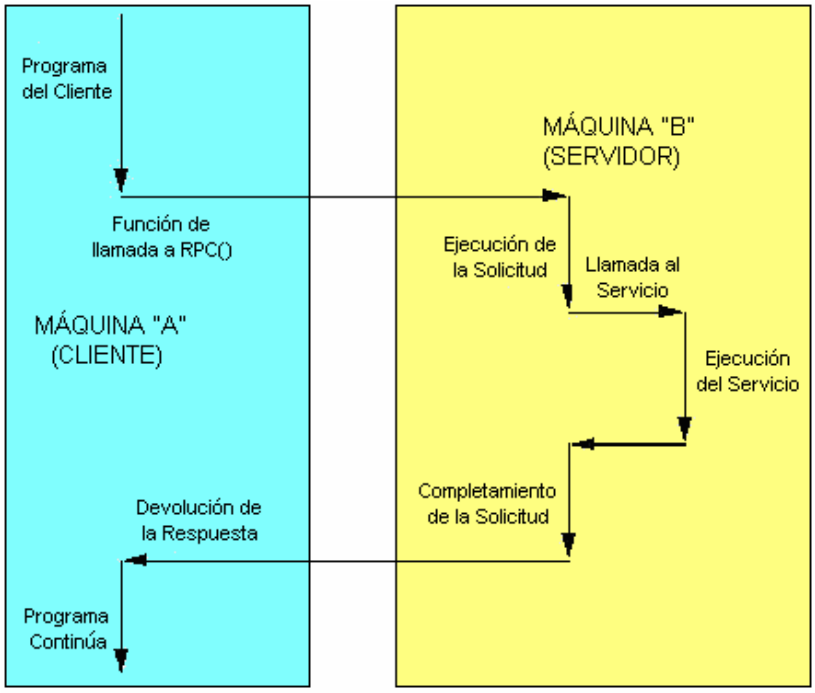

Figura 1.2. Mecanismo de Llamada a Procedimiento Remoto.

1.1.3 Distributed Component Object Model (DCOM).

Distributed Component Object Model (DCOM), en español **Modelo de Objetos de Componentes Distribuidos**, es una tecnología propietaria de Microsoft para desarrollar

componentes software distribuidos sobre varios ordenadores y que se comunican entre sí. Extiende el modelo COM de Microsoft y proporciona el sustrato de comunicación entre la infraestructura del servidor de aplicaciones COM de Microsoft.(Raj 1998)

La adición de la "D" a COM fue debido al uso extensivo de DCE/RPC, o más específicamente la versión mejorada de Microsoft, conocida como MSRPC. En términos de las extensiones que añade a COM, DCOM tenía que resolver los problemas de:

- Aplanamiento - Serializar y deserializar los argumentos y valores de retorno de las llamadas a los métodos "sobre el cable".
- Recolección de basura distribuida, asegurándose que las referencias mantenidas por clientes de las interfaces sean liberadas cuando, por ejemplo, el proceso cliente ha caído o la conexión de red se pierde.

DCOM fue uno de los mayores competidores de CORBA. Los defensores de ambas tecnologías sostenían que algún día serían el modelo de código y servicios sobre Internet. Sin embargo, las dificultades que suponía conseguir que estas tecnologías funcionasen a través de cortafuegos y sobre máquinas inseguras o desconocidas, significó que las peticiones HTTP normales, combinadas con los navegadores *Web* les ganasen la partida. Microsoft, en su momento intentó y fracasó anticiparse a esto añadiendo un transporte extra HTTP a DCE/RPC denominado "ncacn_http" (*Connection-based, over HTTP*).

1.1.4 Common Object Request Broker Architecture (CORBA).

CORBA (Arquitectura para el Envío de Solicitudes de Objetos Comunes) es un *Middleware* o marco de trabajo estándar y abierto de objetos distribuidos que permite a los componentes en la red interoperar en un ambiente común sin importar el lenguaje de desarrollo, sistema operacional, tipo de red, etc. En esta arquitectura, los métodos de un objeto remoto pueden ser invocados "transparentemente" en un ambiente distribuido y heterogéneo a través de un ORB (*Object Request Broker*). Además del objetivo básico de ejecutar simplemente métodos en objetos remotos, CORBA adiciona un conjunto de servicios que amplían las potencialidades de éstos objetos y conforman una infraestructura sólida para el desarrollo de aplicaciones críticas de negocio.(Anónimo 1995)

CORBA es la respuesta del Grupo de Gestión de Objetos (*Object Management Group – OMG*) a la necesidad de interoperabilidad ante la gran proliferación de productos hardware

y software, es decir, permitir a una aplicación comunicarse con otra sin importar el tipo de red, protocolo, sistema operacional o lenguaje de desarrollo.

CORBA automatiza muchas tareas comunes y "pesadas" de programación de redes tales como registro, localización y activación de objetos; manejo de errores y excepciones; codificación y decodificación de parámetros, y protocolo de transmisión.

En un ambiente CORBA, cada Implementación de Objeto, define bien su Interfaz a través una especificación normalizada conocida como IDL (*Interface Definition Language*) a través de la cual en forma Estática (en el momento de compilación) o en forma Dinámica (en el momento de ejecución) un Cliente que requiera el servicio de una Implementación de Objeto, puede ser ejecutada. Las invocaciones a métodos remotos son enviadas por los clientes llamando objetos locales llamados "*Stubs*" (generados por un compilador de IDL - Estático), el cual intercepta dichas invocaciones y continúa el proceso de llevar y retornar automáticamente dicha invocación. La implementación del objeto, no tiene que conocer el mecanismo por el cual un Cliente le ha invocado un servicio.(Zukowsky 2003)

Cuando el Cliente y una Implementación de Objeto están distribuidos por una red, ellos usan el protocolo GIOP/IIOP suministrado por la arquitectura para lograr la comunicación.

La forma de cómo una Implementación de Objeto (desarrollada por un programador de aplicaciones) se conecta a un ORB, es a través de un Adaptador de Objetos. Este adaptador recibe las peticiones por la red e invoca los servicios a la implementación correspondiente.

Actualmente CORBA ya se han resuelto los problemas fundamentales de interoperabilidad y comunicación entre objetos y se han definido y especificado un conjunto de servicios comunes requeridos para la construcción de las aplicaciones, pero donde hay gran actividad es en la especificación de objetos comunes por dominio de aplicación o conocidas en CORBA como Interfaces de Dominio. Allí se trabajan en áreas como Telecomunicaciones, Medicina, Finanzas, Manufactura, etc.

CORBA está fundamentado en dos modelos: Un modelo de Objetos, el cual agrega todas las características de Orientación por Objetos como Tipos de Datos, Abstracción, Polimorfismo y Herencia y un modelo de referencia o arquitectura conocida como OMA (*Object Management Architecture*).

1.1.5 Java/RMI y Java IDL.

Básicamente RMI proporciona la capacidad para llamadas a métodos sobre objetos remotos, los cuales convierten al componente de transporte del objeto en arquitectura de objeto distribuido. También proporciona mecanismos para el registro y persistencia del objeto. Ofrece servicios distribuidos tales como Java IDL que proporciona una forma de conectar, transparentemente, a los clientes Java a los servidores de red utilizando la industria estándar: Lenguaje de Definición de Interfaces (*IDL Interface Definition Language*). Las aplicaciones RMI están, a menudo, compuestas de dos programas separados: un servidor y un cliente. Una aplicación común del servidor crea algunos objetos remotos, realiza referencias para accederlos y se encuentra en espera de que los clientes invoquen los métodos sobre éstos objetos remotos. Una aplicación del cliente tiene una referencia remota a uno o más objetos remotos en el servidor y entonces invoca al método sobre ellos. RMI proporciona los mecanismos a través de los cuales el servidor y el cliente se comunican e intercambian información. Las aplicaciones utilizan uno o dos mecanismos para obtener referencias a objetos remotos. Una aplicación registra sus objetos remotos con la facilidad de denominación del RMI, o bien la aplicación pasa y regresa la referencia a los objetos remotos como parte de su operación normal. Los detalles de comunicación entre objetos remotos está a cargo del RMI; para el programador, la comunicación remota se asemeja a la invocación del método Java. Dado que RMI permite que un solicitante pase objetos a objetos remotos, RMI proporciona los mecanismos necesarios para cargar un código de objeto, así como también de transmitir sus datos. RMI soporta su propio protocolo de transporte denominado Protocolo de Mensajes Remotos Java (*JRMP Java Remote Messaging Protocol*) para definir el conjunto de formatos de mensajes que permiten que los datos pasen a través de una red de computadoras a otra.(Mycrosystems 2010)

La Figura 1.3 representa una aplicación distribuida RMI que utiliza el registro para obtener una referencia a un objeto remoto. El servidor llama al registro para asociar (o ligar) un nombre con un objeto remoto. El cliente busca el objeto remoto por su nombre en el registro del servidor y entonces invoca un método sobre él. La ilustración también muestra que el sistema RMI utiliza un servidor Web para cargar los códigos en bytes de las

clases, del servidor al cliente y del cliente al servidor, para los objetos cuando se les necesita.

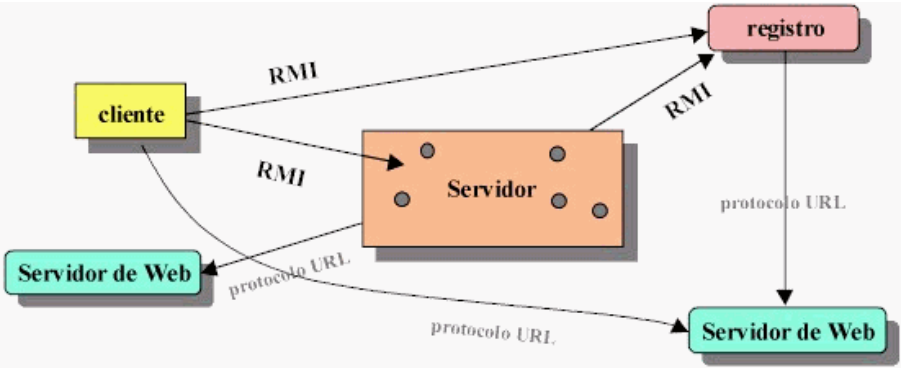

Figura 1.3. Aplicación distribuida RMI.(Herrera 2003)

Java IDL está basado en CORBA. Una característica clave de CORBA es un lenguaje-neutral IDL que soporta la traducción para Java. Para soportar la interacción entre objetos en programas separados, Java IDL proporciona un ORB (*Object Request Broker*). El ORB es una clase biblioteca que facilita la comunicación de bajo nivel entre las aplicaciones Java IDL y otras aplicaciones CORBA.(Zukowsky 2003)

Java RMI es una solución 100% Java para objetos remotos que proporciona todas las ventajas de las capacidades de Java "una vez escrito, ejecutarlo en cualquier parte". Los servidores y clientes desarrollados con Java RMI se muestran en cualquier lugar en la red sobre cualquier plataforma que soporta el ambiente de ejecución de Java. Java IDL, en contraste, está basado en una industria estándar para solicitar, de manera remota, a objetos escritos en cualquier lenguaje de programación. Como resultado, Java IDL proporciona un medio para conectar aplicaciones "transferidas" que aún cubren las necesidades de comercio pero que fueron escritos en otros lenguajes.

Actualmente Java RMI y Java IDL emplean protocolos diferentes para la comunicación entre objetos sobre diferentes plataformas. Java IDL utiliza el protocolo estándar de CORBA IIOP (*Internet Inter-Orb Protocol*). Al igual que IDL, IIOP facilita la residencia de objetos en diversas plataformas y escritos en diversos lenguajes para interactuar de manera estándar. Actualmente Java RMI utiliza el protocolo JRMP (*Java Remote Messaging Protocol*), un protocolo desarrollado específicamente para los objetos remotos de Java.

1.1.6 .Net Remoting.

Tecnología de objetos distribuidos sucesora de DCOM que surgió con el advenimiento de la plataforma .Net y las nuevas tecnologías introducidas por la Microsoft. Tiene como objetivo crear herramientas que faciliten la distribución de la aplicación en red de forma transparente, creando un marco variado y extensible para que los objetos de distintos dominios de aplicaciones, procesos y equipos se puedan comunicar sin problemas. Estas son ideas fundamentales encontradas ya en CORBA o Java RMI, aunque la combinación final es algo diferente. Tiene como características fundamentales:

- Gran flexibilidad.
- Tecnologías independientes del lenguaje de programación y con las comunicaciones en un nivel de abstracción elevado.
- Modelo de programación sencillo y eficaz, así como compatibilidad en tiempo de ejecución que permite interacciones transparentes.
- Va un poco más allá de la distribución de aplicaciones en red de forma transparente.

.*NET Remoting* utiliza una arquitectura flexible y extensible y usa el concepto de .NET llamado Dominio de Aplicación (*AppDomain*) para determinar su actividad. Básicamente un *AppDomain* es el entorno donde se ejecuta la aplicación, por lo que la comunicación se dará entre distintos *AppDomains*, siendo un *AppDomain* el cliente y otro el servidor.(Gil 2002)

Como se muestra en la Figura 1.4 el primer sumidero transforma la invocación del cliente en un mensaje aplanado (serializado) que se irá enviando a cada uno de los sumideros que separan al cliente del sumidero final que introduce la información en el canal real. El mensaje se transmite de un sumidero canal en el cliente a un sumidero canal del servidor, que realiza el proceso inverso al del cliente e invoca el método sobre el objeto indicado. Los parámetros de retorno (si los hay) se envían de vuelta al cliente utilizando el mismo canal. Un sumidero o *sink* es un recipiente capaz de recibir mensajes remotos (*Remoting IMessage*), tratarlos y enviarlos al siguiente sumidero en la cadena de comunicación; y un canal es la forma de ordenar, formatear o transmitir mensajes a través de *AppDomains*, de forma que nosotros podamos decir que queremos transmitir un mensaje

bien por medio de un protocolo de transporte como el TCP o de aplicación como pudiese ser el HTTP o cualquier canal implementado por la arquitectura.(Szimansky 2003)

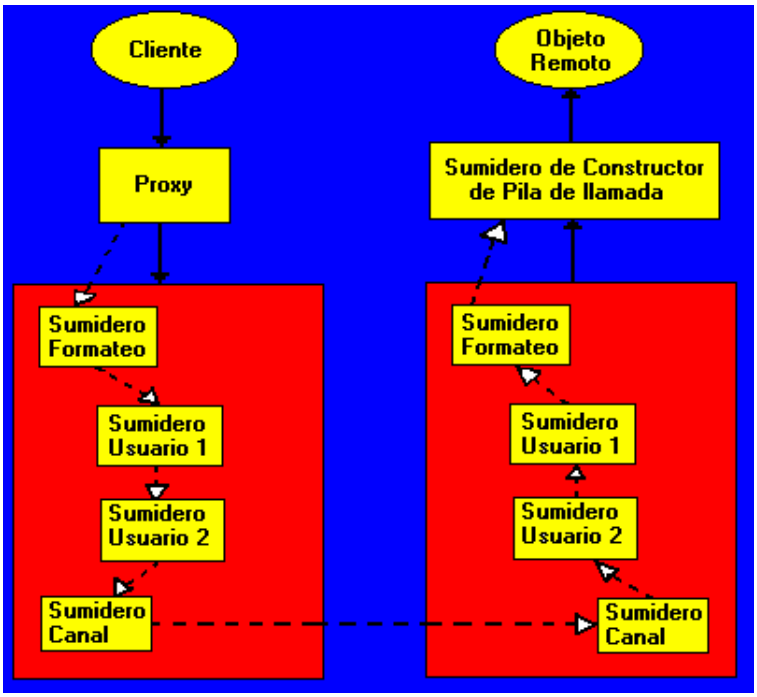

Figura 1.4. Arquitectura *.NET Remoting*

La infraestructura de *.Net Remoting* permite crear dos tipos distintos de objetos:

1. Activados en el servidor (SAO: *Server Activated Object*):
   - ✓ *Singleton*: 1 sola instancia sirve a todos los clientes.
   - ✓ *SingleCall*: 1 instancia por cada solicitud.
2. Activados por el cliente (CAO: *Client Actived Object*):

Los objetos de llamada única (*SingleCall*) se activan cuando llega una petición para ellos, ejecutan la petición y se desactivan. No conservan estado entre conexiones, mientras que los objetos "Singleton“: dan servicio a varias peticiones de clientes diferentes manteniendo el estado.(Gil 2002)

1.1.7 XML-RPC.

XML-RPC es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes. Es un protocolo muy simple ya que sólo define unos cuantos tipos de datos y comandos útiles, además de

una descripción completa de corta extensión. La simplicidad del XML-RPC está en contraste con la mayoría de protocolos RPC que tiene una documentación extensa y requiere considerable soporte de software para su uso.

Fue creado por Dave Winer de la empresa *UserLand Software* en asociación con Microsoft en el año 1998. Al considerar *Microsoft* que era muy simple decidió añadirle funcionalidades, tras las cuales, después de varias etapas de desarrollo, el estándar dejó de ser sencillo y se convirtió en lo que es actualmente conocido como SOAP. Una diferencia fundamental es que en los procedimientos en SOAP los parámetros tienen nombre y no interesan su orden, no siendo así en XML-RPC.

Existen implementaciones para varios sistemas operativos, lenguajes de programación, licencias comerciales y de software libre: C/C++, Delphi, Frontier, Guile, Java, Lisp, Microsoft .NET, Perl, PHP, Python, Real Basic, Rebol, Tcl, WebObjects y Zope. Algunas implementaciones conocidas son:

1. Apache XML-RPC, una implementación en Java.
2. XMLRPC-EPI, una implementación C.
3. XML-RPC-C, una implementación para C y C++.

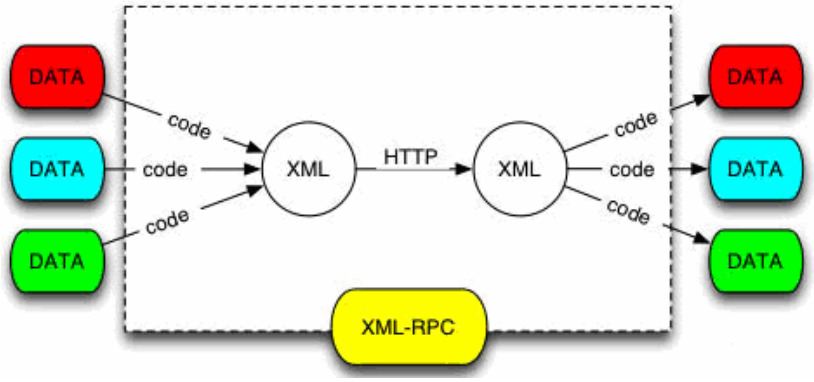

Figura 1.5. Arquitectura XML-RPC.

1.1.8 Web Services.

Un servicio *Web* es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios Web para intercambiar datos en redes de ordenadores como Internet. La

interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.(Erl 2005)

Los servicios Web son un nuevo tipo de aplicaciones Web, modulares, auto-contenidas y auto-descritas, que pueden ser publicadas, localizadas e invocadas a través de la Web. Pueden realizar funciones sencillas como simples llamadas solicitando información o crear y ejecutar complejos procesos de negocio. Una vez que el servicio Web se ha desplegado, este puede ser descubierto e invocado por otras aplicaciones (u otros servicios Web).

Entre las principales ventajas de los servicios Web es que aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen, fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento, además, al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad *firewall* (corta-fuegos) sin necesidad de cambiar las reglas de filtrado. También permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.(Zhang 2007)

Figura 1.6. Descubrimiento e invocación de servicios Web.

SOAP (*Simple Object Access Protocol*, Protocolo Simple de Acceso a Objetos) es un protocolo de mensajes entre computadores. SOAP especifica el formato de mensaje que accede e invoca a los objetos, mas que un objeto en particular. La idea detrás de SOAP es la misma que RPC. También define un protocolo para llamadas a métodos remotos, sin embargo SOAP contiene:

- ✓ Información adicional incluida en el documento XML (lenguaje de marcado extensible), que describe el contenido y como podría ser procesada.
- ✓ Definición de la especificación de algunas estructuras en XML, tales como arreglos.
- ✓ Es un modelo descentralizado, por lo que puede procesarse por varios intermediarios.
- ✓ Características especificas para operaciones clásicas de RPC con parámetros *in/out*, etc.

Los objetivos primordiales de SOAP pueden determinarse así:

a) Establecer un protocolo estándar de invocación de servicios remotos, basado en protocolos estándares de Internet: HTTP (Protocolo de Transporte de Hipertexto) para la transmisión y XML (Lenguaje de Marcado Extensible) para la codificación de datos.

b) Independencia de plataforma, lenguaje de desarrollo e implementación.

El protocolo de comunicación HTTP es el empleado intrínsecamente para la conexión sobre Internet. Garantiza que cualquier cliente con un navegador estándar pueda conectarse con un servidor remoto. La transmisión de datos se empaqueta con XML, que se ha convertido en el estándar del intercambio de datos, salvando las incompatibilidades entre otros protocolos, tales como el NDR (*Network Data Representation*) o el CDR (*Common Data Representation*).

WSDL son las siglas de *Web Services Description Language*, un formato XML que se utiliza para describir servicios Web. La versión 1.0 fue la primera recomendación por parte del W3C y la versión 1.1 no alcanzó nunca tal estatus. La versión 2.0 se convirtió en la recomendación actual por parte de dicha entidad.(Chase 2006 a)

WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las

operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

UDDI son las siglas del catálogo de negocios de Internet denominado *Universal Description, Discovery and Integration*. El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el contexto de los servicios Web. El registro de un negocio en UDDI tiene tres partes:

- Páginas blancas - dirección, contacto y otros identificadores conocidos.
- Páginas amarillas - categorización industrial basada en taxonomías.
- Páginas verdes - información técnica sobre los servicios que aportan las propias empresas.

UDDI es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.(Chase 2006 b)

1.2 Arquitectura Orientada a Servicios (SOA).

La capacidad para responder rápidamente ante los cambios y optimizar los procesos de negocio es un factor clave para la competitividad y el crecimiento de las organizaciones. La agilidad de éstas puede verse cuestionada si se apoya en entornos de IT (Tecnologías de Información) que no pueden responder de forma flexible a los cambios que afectan a la actividad de negocio. Liberar el potencial que poseen las aplicaciones y recursos de IT y hacerlo disponible de forma general a toda la organización facilita la optimización de procesos y mejora la agilidad empresarial. La Arquitectura Orientada a Servicios (*SOA, Service Oriented Architecture*) es una filosofía de diseño que permite un mejor alineamiento de las Tecnologías de Información (IT) con las necesidades de negocio, permitiendo a empleados, clientes y socios comerciales responder de forma más rápida y adaptarse adecuadamente a las presiones del mercado.(Microsoft 2006)

1.2.1 Evolución de SOA.

Breve historia de XML.

Como el HTML, el *Extensible Markup Language* (XML) fue una creación de la W3C derivada del *Standard Generalized Markup Language* (SGLM) que ha existido desde finales de los 60s. Este metalenguaje ampliamente usado permitió a las organizaciones añadir inteligencia a datos crudos del documento.

XML ganó popularidad durante el movimiento de *eBusiness* de finales de los 90s, donde lenguajes de encriptación del lado del servidor conducen negocios por vía Internet. A través del uso de XML, los desarrolladores pudieron adjuntar el significado y el contexto para cualquier información transmitida a través de los Protocolos de Internet.

No sólo fue XML usado para representar datos en una manera estándar, el lenguaje mismo fue utilizado como la base para una serie de especificaciones adicionales. El XML *Schema Definition Language* (XSD) y el XSL *Transformation Language* (XSLT) fueron ambos publicados usando XML. Estas especificaciones, de hecho, se han convertido en partes cruciales del núcleo del conjunto tecnológico XML.

La arquitectura de representación de datos XML constituye el estrato de la fundación de SOA. Dentro de eso, XML establece el formato y estructura de los mensajes que viajan entre los servicios. Los esquemas XSD conservan la integridad y la validez de datos del mensaje, y XSLT es empleado para posibilitar la comunicación entre representaciones dispares de datos a través del mapeo del esquema. En otras palabras, usted no puede dar un paso dentro de SOA sin involucrarse con XML.(Erl 2005)

Breve historia de Servicios Web.

En 2000, el W3C recibió una sumisión para la especificación *Simple Object Access Protocol* (SOAP). Esta especificación fue originalmente diseñada para unificar (y en algunos casos reemplazar) la comunicación RPC propietaria. La idea era para parametrizar datos transmitidos entre componentes a ser serializados a XML, transportados, y luego deserializados de vuelta a su formato original.

La parte más importante de un servicio Web es su interfaz pública. Es la parte central de la información que le asigna al servicio una identidad y posibilita su invocación. Por consiguiente, una de las primeras iniciativas en apoyo de los servicios Web fue el *Web Service Description Language* (WSDL). El W3C recibió la primera sumisión del lenguaje WSDL en 2001 y desde entonces ha continuado revisando esta especificación.

Para fomentar la visión de interoperabilidad abierta, los servicios Web requirieron un formato de comunicaciones amigable a la Internet. Aunque las alternativas, como XML-RPC, fueron consideradas, SOAP ganó como el favorito de la industria y queda como el primer estándar de envío de mensajes para el uso con servicios Web.

En apoyo al nuevo papel de SOAP, el W3C respondió realizando versiones más nuevas de la especificación para tener en cuenta ambos, el estilo RPC (*RPC-style*) y el mensaje de estilo de documento (*document-style*). Eventualmente, la palabra "SOAP" se consideró ya no un acrónimo para "*Simple Object Access Protocol*". A partir de versión 1.2 de la especificación, se convirtió en un término auto sostenible.

Completando la primera generación de la familia de normas de los servicios Web estuvo la especificación UDDI. A diferencia de WSDL y SOAP, UDDI aún no ha logrado aceptación en la industria, y queda como una extensión optativa para SOA.

Los "*custom Web services*" fueron desarrollados para acomodar una colección variada de requisitos de negocio especializados, y un mercado de terceros emergió promoviendo diversas utilidades de los servicios para la venta o arrendamiento.

Breve historia de SOA.

No pasó mucho tiempo antes de que las organizaciones comenzasen a darse cuenta de que en lugar de acomodar aplicaciones simplemente distribuidas ya existentes, los servicios Web podrían convertirse en la base de una plataforma arquitectónica separada que podría apoyar los beneficios de la tecnología de los servicios Web dispuestos a percatarse del concepto de servicios en la empresa. Así, la arquitectura orientada a servicios introdujo el pensamiento prevaleciente de las tecnologías de la información (IT).

En este punto una arquitectura orientada a servicios estaba frecuentemente clasificada de diferentes maneras, a menudo a merced de la tecnología de implementación usada para construir los servicios. Un modelo antiguo, en su mayor parte inspirado por el conjunto inicial de estándares de servicios Web, definió a SOA como una arquitectura modelada alrededor de tres componentes básicos: el buscador de servicio, el proveedor de servicios, y el registro de servicio (Figura 1.7).

Los estándares de la primera generación de servicios Web cumplieron este modelo como sigue:

- WSDL describió el servicio.
- SOAP proveyó el formato de envío de mensajes usado por el servicio y su buscador.
- UDDI proveyó el formato estándar del registro de servicios.

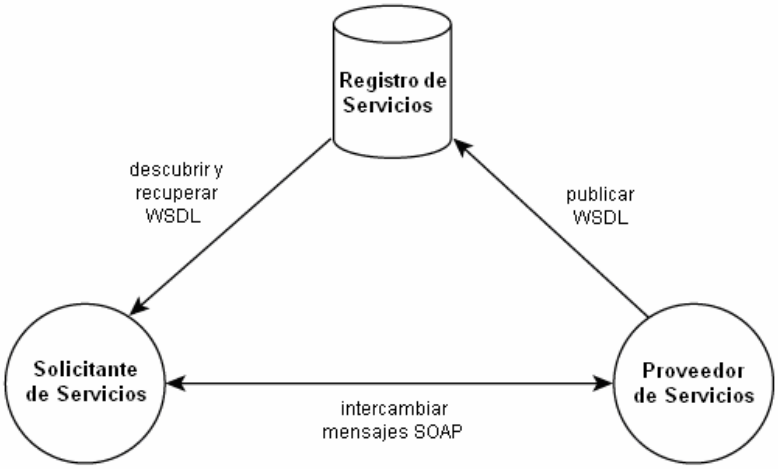

Figura 1.7. Un antiguo modelo de SOA.

De una perspectiva física de arquitectura, esta primera variación de una arquitectura orientada a servicio basada en servicios Web realmente va más allá del modelo primitivo SOA. Usted puede decir que SOA primitiva no precisó el uso de un registro de servicio. En lugar de eso, el "descubrimiento" está clasificado como una de las características del SOA contemporáneo y es promovido, en un nivel de servicios, a través de los principios de orientación a servicios.

Numerosas características contemporáneas de SOA son el resultado del agresivo desarrollo y las iniciativas colaborativas que ha producido una serie de extensiones para la primera generación de la plataforma de servicios Web. La segunda generación o especificaciones "WS-*", son extensiones que están dirigidas a áreas específicas de funcionabilidad con la meta global de elevar la plataforma de tecnología de servicios Web a nivel a empresarial. WS-* fue también un interés el aplicar conceptos de orientación a servicios para el mundo del análisis de negocios. A través de la orientación a servicios, la lógica del negocio puede ser limpiamente encapsulada y abstraída de la tecnología de automatización subyacente. Esta visión ha ido más allá soportada por el alza de los

lenguajes de definición de procesos del negocio, más notablemente WS-BPEL. Estos no sólo tuvieron en cuenta la descomposición tradicional del modelado de los procesos de negocio en una serie de servicios, sino que unieron la brecha entre análisis e implementación, proveyendo un lenguaje capaz de expresar completamente la lógica del negocio en un formato concreto y ejecutable.

SOA es verdaderamente una evolución. Su prominencia hoy es el resultado de numerosas e interrelacionadas iniciativas conducidas por una variedad de organizaciones de estándares y los vendedores de software. A través de un ambiente volátil que se alimentó por una mezcla de colaboración y competición, las extensiones son estratégicamente situadas, cada una definiendo una parte específica de lo que llamamos la plataforma tecnológica de SOA contemporánea.(Krafzig, Banke et al. 2005)

1.2.2 Principios de la orientación a servicios.

1. Contratos de servicios: Los servicios se adhieren a acuerdos de comunicación.
2. Autonomía: Los servicios tienen control sobre la lógica que encapsulan.
3. Acople bajo: Los servicios mantienen relaciones con dependencias bajas.
4. Abstracción: Los servicios ocultan la lógica al mundo exterior, solo se muestra el contrato.
5. Reutilización: La lógica se divide en servicios con la intención de promover su reutilización.
6. Composición: Colecciones de servicios pueden ser coordinadas y ensambladas en composiciones de servicios.
7. Ausencia de estado: Los servicios minimizan el mantenimiento de información sobre una actividad específica.
8. Descubrimiento: Los servicios se diseñan de modo que puedan ser descubiertos a través de mecanismos de descubrimiento.

1.2.3 Beneficios de SOA.

Los beneficios de SOA para una organización se plasman a dos niveles distintos: al del usuario corporativo y a nivel de la organización de IT. Desde el punto de vista de la empresa, SOA permite el desarrollo de una nueva generación de aplicaciones dinámicas

que resuelven una gran cantidad de problemas de alto nivel, fundamentales para el crecimiento y la competitividad. Las soluciones SOA permiten entre otras cosas:

- **Mejorar la toma de decisiones.** Al integrar el acceso a los servicios e información de negocio dentro de un conjunto de aplicaciones dinámicas compuestas, los directivos disponen de más información y de mejor calidad (más exacta y actualizada).
- **Mejorar la productividad de los empleados.** Un acceso óptimo a los sistemas y la información y la posibilidad de mejorar los procesos permiten a las empresas aumentar la productividad individual de los empleados.
- **Potenciar las relaciones con clientes y proveedores**. Las ventajas de SOA trascienden las fronteras de la organización. Los procesos de fusión y compra de empresas se hacen más rentables al ser más sencilla la integración de sistemas y aplicaciones diferentes.

SOA contribuye también a documentar el modelo de negocio de la empresa y a utilizar el modelo de negocio documentado para integrar en él y dar respuesta a las dinámicas de cambio que se produzcan y optimizarlo de acuerdo con ellas. Desde el punto de vista de los departamentos de IT, la orientación a servicios supone un marco conceptual mediante el cual se puede simplificar la creación y mantenimiento de sistemas y aplicaciones integradas, y una fórmula para alinear los recursos de IT con el modelo de negocio y las necesidades y dinámicas de cambio que le afectan.(Microsoft 2006)

- ✓ **Aplicaciones más productivas y flexibles.** La estrategia de orientación a servicios permite a IT conseguir una mayor productividad de los recursos de IT existentes –como pueden ser las aplicaciones y sistemas ya instalados- y obtener mayor valor de ellos de cara a la organización sin necesidad de aplicar soluciones de integración desarrolladas ex profeso para este fin.
- ✓ **Desarrollo de aplicaciones más rápido y económico**. El diseño de servicios basado en estándares facilita la creación de un repositorio de servicios reutilizables que se pueden combinar en servicios de mayor nivel y aplicaciones compuestas en respuesta a nuevas necesidades de la empresa.
- ✓ **Aplicaciones más seguras y manejables**. Las soluciones orientadas a servicios proporcionan una infraestructura común para desarrollar servicios seguros, predecibles

y manejables. Conforme van evolucionando las necesidades de negocio, SOA facilita la posibilidad de añadir nuevos servicios y funcionalidades para gestionar los procesos de negocio críticos. Se accede a los servicios y no a las aplicaciones, y gracias a ello la arquitectura orientada a servicios optimiza las inversiones realizadas en IT potenciando la capacidad de introducir nuevas capacidades y mejoras.

1.2.4 Cómo se resuelven los retos de SOA.

Embarcarse en un proyecto de SOA supone tener que resolver una serie de retos, tanto a nivel organizativo como técnico, y estos retos pueden convertirse en verdaderas barreras insuperables si se ha partido de la idea de que SOA es el remedio para toda clase de males. Para que las iniciativas de adopción de SOA tengan un fin satisfactorio, hay que asegurarse de que se cumplen una serie de condiciones indispensables:

- **Definir claramente los objetivos de negocio**. El primer paso a la hora de adoptar SOA es identificar con claridad los problemas o retos empresariales más prioritarios. Cuando más precisa sea esa formulación, más fácilmente se podrá delimitar la dirección y el alcance de cualquier proyecto SOA.
- **Definir claramente el alcance del proyecto SOA**. El objetivo real de cada iniciativa SOA debe ser responder a necesidades concretas de negocio y crear soluciones en pasos discretos, incrementales e iterativos.
- **Evitar introducir SOA sin motivos reales que lo justifiquen**. La adopción de SOA no debe considerarse una necesidad tecnológica, sino organizativa: debe responder a las necesidades de la organización.
- **Gestionar el proceso**. Los servicios y aplicaciones se corresponden con procesos y los reportes de información deseados a través de las diversas áreas funcionales de la organización.

1.2.5 Capas de servicios.

La capa de interfaz de servicio está ubicada entre el proceso de negocio y la capa de aplicación. Ahí es donde la conectividad de servicio reside y es por consiguiente el área de una empresa en donde las características de SOA son más prevalecientes. Específicamente, se necesita contestar a las siguientes preguntas:

¿Qué lógica debería ser representada por los servicios?

La lógica de la empresa puede estar dividida en dos dominios primarios: la lógica del negocio y la lógica de la aplicación. Los servicios pueden ser modelados para representar ya sea o ambos tipos de lógica, tanto como los principios de orientación a servicios pueden ser aplicados. Debe establecerse una relación acoplada ligeramente entre la lógica del negocio y la lógica de aplicación

¿Cómo los servicios deberían guardar relación con la lógica de la aplicación existente?

Mucho de esto depende de si la lógica de la aplicación existente necesita estar al descubierto vía servicios o ya sea que la lógica nueva está siendo desarrollada en soporte a los servicios. Los sistemas existentes pueden imponer cualquier número de restricciones, limitaciones, y requisitos que necesitan ser tomados en consideración durante el diseño de servicio. Los servicios están diseñados específicamente para representar la lógica de la aplicación y deben existir en capas separadas.

¿Cómo pueden los servicios representar mejor la lógica del proceso de negocio?

La lógica del negocio está definida dentro de los modelos y procesos de negocio de una organización. Cuando los servicios modelados representen la lógica del negocio, es muy importante asegurar que la representación de servicio de esta lógica esté alineada con los modelos de negocio existentes.

¿Cómo pueden ser los servicios fortalecidos y situados para promocionar agilidad?

La llave para crear SOA ágil está en minimizar las dependencias que cada servicio tiene dentro de su propio procesamiento de la lógica. Los servicios que contienen reglas de negocio están obligados a implementar y actuar sobre estas reglas en tiempo de ejecución. Esto limita la habilidad del servicio para ser utilizado fuera de ambientes que requieren estas reglas. De modo semejante, los servicios controladores que están incrustados con la lógica requerida para componer otros servicios pueden desarrollar dependencias en la estructura de composición. Al introducir una capa controladora padre encima de capas de servicio más especializadas se logra establecer una posición centralizada para las reglas de negocio y la lógica relacionada a la secuencia en la cual los servicios son ejecutados.(Microsoft 2006)

Las tres capas primarias de abstracción que se identifican para SOA son:

- La capa de servicios de aplicación.
- La capa de servicios de negocio.
- La capa de servicios de orquestación.

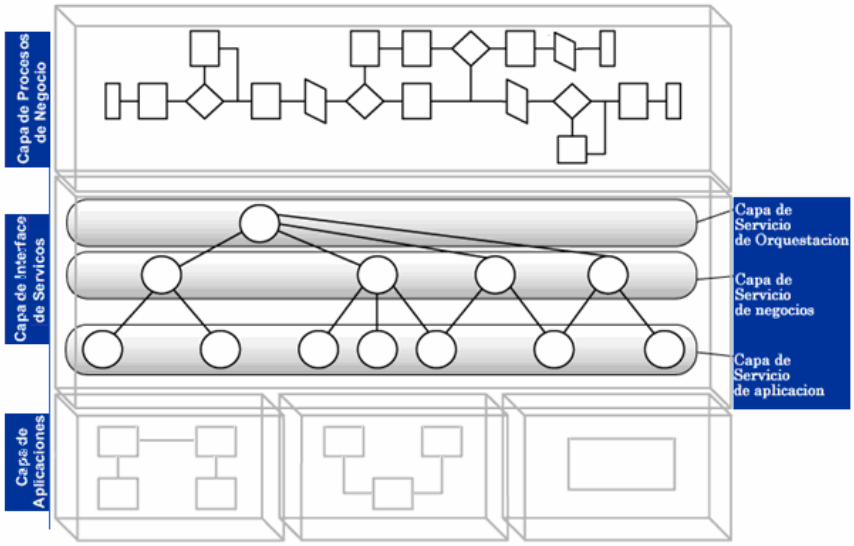

Figura 1.8. Las tres capas primarias de servicios.

La capa de servicios de aplicación:

La capa de servicio de aplicación establece los diferentes niveles que existen para expresar el funcionamiento de una tecnología específica. Los servicios que residen dentro de esta capa simplemente pueden ser referidos como los servicios de aplicación. Su propósito es proporcionar funciones reutilizables relacionadas con el procesamiento de los datos dentro de nuevo o ambientes de aplicación. Los servicios de aplicación normalmente tienen las características siguientes:

- Exponen las funcionalidades dentro de un contexto del proceso específico.
- Utilizan los recursos disponibles dentro de una plataforma dada.
- Son soluciones agnósticas.
- Son genéricos y reutilizables.
- Pueden usarse para lograr la integración del punto a punto con otros servicios de la aplicación.
- Pueden consistir en una mezcla de servicios desarrollados dentro de la empresa y servicios de terceros que se han comprado o se han arrendado.

La capa de servicios de negocio:

Mientras los servicios de la aplicación son los responsables de representar tecnología y lógica de una aplicación, la capa de servicios de negocio introduce un servicio que se interesa solamente con representar la lógica de negocio, nombrado servicio de negocio. La abstracción de capa de servicio de negocio incluye la creación de dos modelos de servicio de negocio adicionales:

- ✓ Servicio de negocios centrado en tareas: Es un servicio que encapsula la lógica de negocio específica a una tarea o el proceso de negocio. Este tipo de servicio generalmente se requiere cuando la lógica del proceso de negocio no se centraliza como parte de una capa de orquestación. Pero limitan la reutilización.(Erl 2005)
- ✓ Servicio de negocios centrado en entidades: Es un servicio que encapsula una entidad específica de negocio. Son útiles para servicios agnósticos, altamente reutilizables.

La capa de servicios de orquestación:

La capa de servicios de orquestación es la de mayor nivel de abstracción y se encarga de manejar los detalles de la interacción entre los demás servicios exigiendo y garantizando que el funcionamiento se ejecute en una sucesión específica. Dentro de la capa de servicios de orquestación, los servicios del proceso componen otros servicios que proporcionan juegos específicos de funciones, independientes de las reglas de negocio. Esta capa permite enlazar la lógica del proceso a las interacciones de servicios dentro de la lógica del flujo de trabajo. Lo anterior combina la modelación del proceso de negocio con el diseño y modelación de arquitecturas orientadas a servicios (SOA).(Bell 2008)

1.2.6 Estrategias de desarrollo.

La estrategia debe basarse en las prioridades de una organización para establecer el balance correcto entre la entrega de metas de migración de largo plazo con el cumplimiento de requisitos de corto plazo. Tres estrategias comunes han emergido, cada una ocupándose de este problema de una manera diferente.

De arriba hacia abajo.

La estrategia de arriba hacia abajo soporta la creación de las tres capas de servicios. Es común que este acercamiento resulte en la creación de numerosos negocios reutilizables y

aplicaciones de servicios. La estrategia de arriba hacia abajo típicamente contendrá una cierta cantidad o todos los pasos ilustrados y descritos en la Figura 1.9. Este proceso da por supuesto que los requisitos del negocio ya han sido coleccionados y definidos.

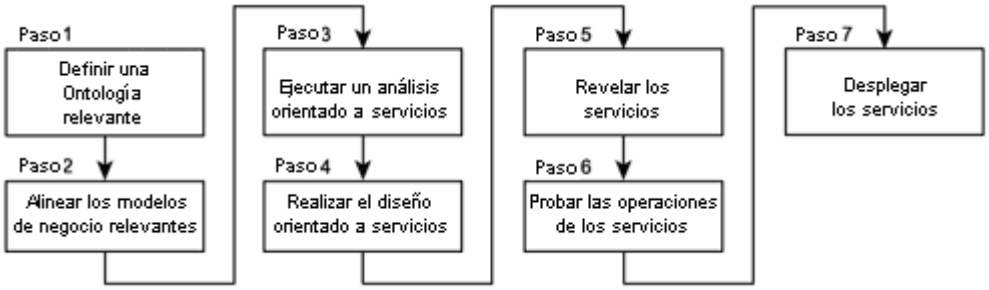

Figura 1.9. Estrategia de desarrollo de arriba hacia abajo.

El enfoque de arriba hacia abajo en la construcción de SOA generalmente resulta en una arquitectura de servicios de alta calidad. El diseño y los parámetros en torno a cada uno de los servicios se analizan a fondo, la maximización de la reutilización potencial y las oportunidades para optimizar las composiciones. Los obstáculos a raíz de un enfoque de arriba hacia abajo por lo general están asociados con el tiempo y dinero. Las organizaciones están obligadas a invertir significativamente en el análisis inicial de los proyectos que pueden tener una gran cantidad de tiempo sin resultados inmediatos.(Microsoft 2006)

De abajo hacia arriba.

Este acercamiento esencialmente promueve la creación de servicios como una manera de satisfacer los requisitos de la aplicación. Los servicios Web son implementados "como sean necesarios" y modelados para encapsular la lógica de la aplicación sirviendo mejor a los requisitos inmediatos de la solución. La integración es la motivadora primaria para diseños de abajo hacia arriba, dónde la necesidad de tomar ventaja del marco de trabajo abierto de comunicaciones SOAP puede ser encontrada en el simple hecho de anexar servicios como envolturas para sistemas legados. Un típico enfoque de abajo hacia arriba sigue un proceso parecido a lo uno explicado en la Figura 1.10. Este proceso da por supuesto que los requisitos del negocio ya han sido coleccionados y definidos.

La mayoría de las organizaciones que actualmente están construyendo los servicios Web aplican el enfoque de abajo hacia arriba. La razón principal detrás de esto es que las organizaciones simplemente pueden añadir servicios Web a sus entornos de aplicaciones

existentes para aprovechar esta tecnología. La arquitectura de servicios Web y los servicios de orientación en el que se añaden se mantiene sin cambios, por lo tanto, los principios son raramente tenidos en cuenta.

Aunque el diseño de abajo hacia arriba permite la creación eficiente de los servicios Web requerido por las aplicaciones, la implantación de una arquitectura SOA en un momento dado puede necesitar una gran cantidad de reajustes, o incluso la introducción de nuevas capas de servicios.

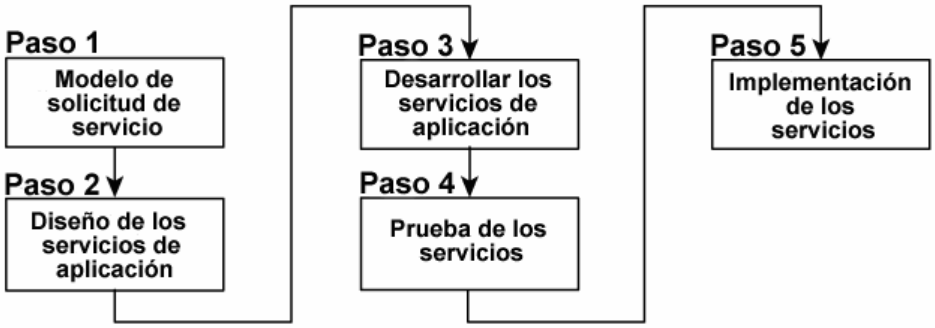

Figura 1.10. Estrategia de desarrollo de abajo hacia arriba.

Ágil.

Para muchas organizaciones es útil mirar las dos estrategias anteriores como extremas y encontrar un adecuado intermedio. Esto es posible definiendo un nuevo proceso que tiene en cuenta el análisis de los niveles de negocio ocurriendo concurrentemente con diseño de servicio y desarrollo. También conocido como *satisfacer-en-el-medio*, la estrategia ágil es más complicada que las dos previas, simplemente porque necesita satisfacer dos conjuntos opuestos de requisitos. Los pasos del proceso se muestran en la Figura 1.11, demuestran un ejemplo de cómo una estrategia ágil puede usarse para alcanzar las metas de los enfoques de arriba hacia abajo y de abajo hacia arriba respectivamente.

Esta estrategia toma lo mejor de las dos anteriores, sin poner en peligro la integridad del modelo de negocio de una organización y los servicios orientados a las características de la arquitectura.

A pesar de que cumple tanto a corto como a largo plazo, el resultado neto del empleo de esta estrategia es aumentar los esfuerzos asociados a la prestación de cada servicio.

Este enfoque impone tareas de mantenimiento que se requieran para garantizar que los servicios existentes se mantengan en consonancia con los modelos de negocio revisados. Incluso con un proceso de mantenimiento en su lugar, es difícil lograr la alineación con un modelo de negocio que se encuentra en constante cambio.(Erl 2005)

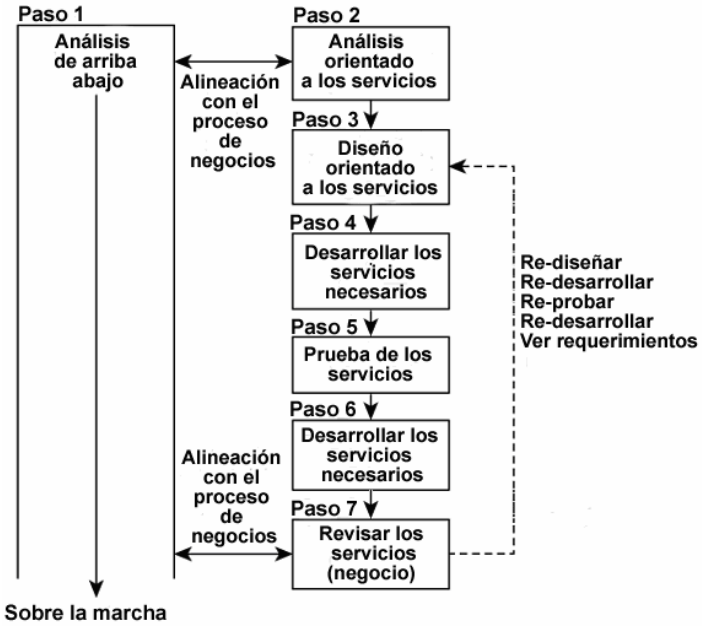

Figura 1.11. Estrategia de desarrollo ágil.

1.3 SOA y la gestión de procesos de negocios (BPM).

El concepto de BPM (*Business Process Management*) está también muy ligado a SOA. BPM es una disciplina de gestión que combina una visión centrada en procesos y de integración de funcionalidades que pretende mejorar la efectividad de las organizaciones. Una solución BPM dispone de los medios necesarios para la realización efectiva de estos procesos así como las funcionalidades necesarias para que los gestores de las empresas puedan controlar y modificar los flujos de trabajo tanto manuales como automáticos.

La gestión de procesos de negocio tiene sus orígenes en los Sistemas de Gestión de Calidad Total y la reingeniería de procesos. Puesto que les añade un marco tecnológico de desarrollo, BPM es más que una combinación de estas disciplinas: BPM es una disciplina de gestión de procesos dirigida mediante Tecnologías de Información, capaz de mejorar la agilidad organizativa y que mejora la capacidad de las personas para introducir cambios en los procesos e innovar de forma rápida. Por consiguiente, BPM permite el alineamiento de

las tecnologías de información con las actividades de negocio, tanto en el seno de la propia organización como fuera de ella, con socios comerciales, proveedores y clientes.

Los procesos de negocio pueden ser estructurados o no estructurados, dependiendo de hasta qué punto los pasos que comprenden son pasos bien establecidos –y susceptibles, por tanto, de automatización- o intercambiables, y generalmente ejecutados por personas solamente o por personas que interactúan con sistemas. Las personas son una parte esencial de prácticamente cualquier proceso de negocio: aplican las soluciones y disponen de la visión que hace avanzar a una empresa, por lo que el objetivo debe ser aumentar su capacidad para crear e innovar y ser más productivas (y no hacer reingeniería pretendiendo colocar a las personas fuera de los procesos).

Aunque BPM puede considerarse como una entidad al margen de las iniciativas SOA, la capacidad para definir nuevos procesos de negocio de forma flexible y rápida es mucho mayor si los recursos de los sistemas de IT se exponen en la forma de orientación a servicios.

1.4. Conclusiones Parciales

Con el estudio realizado queda evidenciado la importancia de acometer las soluciones computacionales de diferentes procesos de gestión de información empresarial a través de estas nuevas tecnologías agrupadas bajo el concepto de Arquitectura Orientada a los Servicios que tiende a la reutilización de servicios en aquellos procesos, y aprovechar al máximo las capacidades de Tecnologías de Información desplegadas como en nuestra Universidad que cuenta con varias redes locales interconectadas en una intranet.

La gestión de la información de postgrado debe realizarse en diferentes áreas del centro (facultades y centros de estudio o de investigación), pero a su vez deben ser accedidas por otras áreas como la Dirección de Postgrado. El diseño e implementación de un sistema automatizado con este fin bajo los preceptos de SOA sería muy favorable, eficiente y mantenible, pero también los servicios definidos para el mismo o parte de este, pueden ser aprovechados por otros sistemas de información de la propia entidad.

La estrategia de desarrollo SOA a utilizar en este trabajo será la estrategia Ágil o Satisfacer-en-el-medio, por considerar la más adecuada al proyecto.

CAPÍTULO 2. ANÁLISIS ORIENTADO A SERVICIOS DEL PROCESO DE MAESTRÍAS Y ESPECIALIDADES.

CAPÍTULO 2. ANÁLISIS ORIENTADO A SERVICIOS DEL PROCESO DE MAESTRÍAS Y ESPECIALIDADES.

Los procedimientos del Proceso de Maestrías y Especialidades consisten en un conjunto estructurado de tareas o actividades estrechamente relacionadas entre sí, y el objetivo que persiguen es formar másteres y especialistas. La organización donde tienen lugar estas actividades o tareas es el ámbito de la educación de postgrado en nuestras instituciones calificadas para ello y sus canales correspondientes tales como: facultades y sus departamentos docentes y centros de estudio o investigación, la dirección de postgrado de la UCLV y la secretaría general de la UCLV y las secretarías de las áreas, entre otros.

Este trabajo se centra en los siguientes procedimientos:

- Procedimiento de gestación y aprobación de nuevos programas de Maestrías y Especialidades.
- Procedimiento de inicio, matrícula y ejecución y control de Ediciones de Programas de Maestrías y Especialidades.
  - Sub-procedimiento de planificación de inicio de Ediciones de Programas de Maestrías y Especialidades.
  - Sub-procedimiento oficial de inicio de Ediciones de Programas de Maestrías y Especialidades.
- Procedimiento de ejecución y control en las áreas de Maestrías y Especialidades.

En este capítulo se comienza la fase de análisis orientado a servicios y se decide aplicar una estrategia ágil pues esta es la que mejor se adapta a las características del proyecto, donde en poco tiempo se debe implementar una arquitectura orientada a servicios en paralelo con el descubrimiento del modelo de negocios del problema en cuestión.

2.1 Análisis orientado a servicios.

El proceso de determinar cómo la automatización de los requerimientos del negocio pueden ser representados a través de la orientación a servicios es el dominio del análisis orientado a servicios.

En cualquier estrategia de desarrollo el análisis orientado a servicios juega un papel fundamental, como se muestra en la figura 2.1, por lo que en este epígrafe profundizaremos en su estudio.

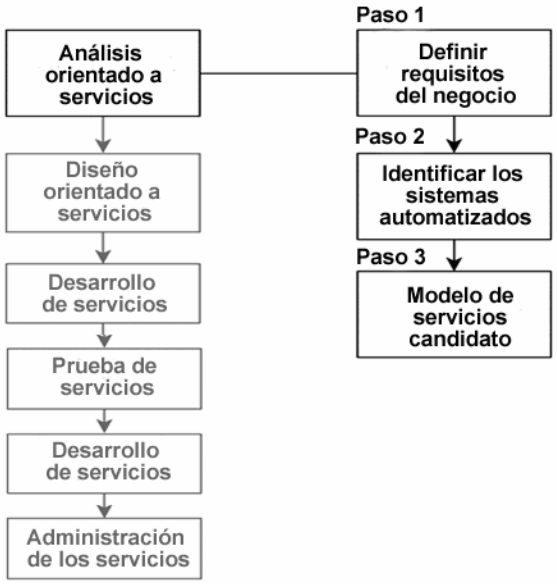

Figura 2.1. Proceso de análisis orientado a servicios.

Durante esta fase lo primero que debemos preguntarnos es:

- ✓ ¿Qué servicios necesitan ser construidos?
- ✓ ¿Qué lógica debería ser encapsulada por cada servicio?

La extensión en la cual estas preguntas son respondidas está directamente relacionada con la cantidad de esfuerzo invertido en el análisis. Las metas globales al realizar un análisis orientado a servicio son las siguientes:

- ✓ Definir un conjunto preliminar de operaciones candidatas de servicios.
- ✓ Agrupar las operaciones candidatas de servicios en contextos lógicos. Estos contextos representan servicios candidatos.
- ✓ Definir los límites preliminares de los servicios a fin de que no se recubran con cualquier servicio existente o planeado.
- ✓ Identificar la lógica encapsulada con potencial de reutilización.
- ✓ Asegurar que el contexto de la lógica encapsulada es apropiado para su pretendido uso.
- ✓ Definir cualquier modelo preliminar de composición de servicios conocido.

A continuación se analizan una serie de pasos que se deben tener en cuenta durante el análisis orientado a servicios.

Paso 1: Definir requerimientos del negocio.

Para este paso se requiere de la documentación referente a los requerimientos del negocio, siendo de mucha utilidad un diagrama del proceso expresado en notación de modelado de proceso de negocio (BPMN, *Business Process Modeling Notation*)

Dado que el alcance de los análisis se centra alrededor de la creación de servicios en apoyo de una solución orientada a servicio, sólo los requisitos relacionados con el alcance de esa solución deberían ser considerados. Los requisitos del negocio deben ser lo suficientemente maduros para que se pueda definir un proceso de automatización de alto nivel. Esta documentación del proceso de negocio será usada como el punto de partida del proceso de modelación de servicios que se descubren en el Paso 3.

Descripción de los proceso de negocio a automatizar.

1. **Procedimiento de gestación y aprobación de nuevos programas de Maestrías y Especialidades.**

Lo primero que ocurre es que surge una nueva necesidad de un programa PGA por parte del Centro de Producción o Servicios o surge una nueva propuesta de Programa de PGA por Profesores de un Departamento. Esta solicitud o propuesta es analizada por el departamento correspondiente en la entidad. Si el PGA es una especialidad de postgrado, el departamento tiene que consultar al OACE correspondiente, el cual en caso de aprobación propone especialistas para el Comité Académico (CA) y el Claustro. El departamento correspondiente se encarga de proponer al decano o director la composición de ese CA, si no es aprobada la propuesta, la misma es retornada al departamento. Después de aprobación (puede ser en Consejo de Dirección o Consejo Científico) el decano o director propone la composición del CA al rector, quien analiza la propuesta y en caso de ser aprobado el CA emite una Resolución Rectoral que certifica la autorización a ese CA. En caso de no aprobar la propuesta se retorna al decano o director correspondiente.

Una vez que ha sido aprobada la constitución del CA, este trabaja en la redacción del Programa de PGA y lo presenta al Consejo Científico del área correspondiente, quien es el

encargado de analizarlo y aprobarlo. Si el programa no es aprobado se trabaja nuevamente en su redacción para presentarlo nuevamente al Consejo Científico. Si el Programa es aprobado entonces el CA presenta el Programa a la COPEP-UCLV., la cual la remite a una comisión *ad-hoc* y esta la analiza y presenta sus recomendaciones en la reunión mensual de ese órgano, incluyendo su aprobación. Si no es aprobado, el programa se regresa de nuevo al CA**.** Si es aprobado se emite un Dictamen de COPEP-UCLV y la dirección de PG remite ese Dictamen y el programa aprobado por COPEP-UCLV a la COPEP-MES, organismo que entrega a una Comisión correspondiente el programa la cual lo analiza y propone o no aprobación definitiva. Si no es aprobado, el programa se regresa de nuevo al CA. Si se aprueba se genera en el sistema automatizado y se transforma a formato XML y se valida para su posterior uso, la COPEP-MES emite Dictamen de aprobación y remite al Ministro, el cual emite una Resolución Ministerial que permite la ejecución del Programa PGA dado (Figura 2.2).

2. <u>Procedimiento de inicio, matrícula y ejecución y control de Ediciones de Programas de Maestrías y Especialidades</u>.

a. *Sub-procedimiento de planificación de inicio de Ediciones de Programas de Maestrías y Especialidades*.

Al surgir una nueva solicitud de un Centro de Producción o Servicios de Edición de PGA o al surgir una propuesta de Edición de PGA por Comité Académico, el departamento correspondiente analiza la solicitud o propuesta y si está de acuerdo, propone la apertura del PGA en su Plan Postgrado Anual (de departamento). Si el Consejo de Dirección de Facultad aprueba el Plan Postgrado Anual propuesto por su VDPG (condensando los planes propuestos por sus departamentos) entonces el VDPG entrega a la DPG su Plan Postgrado Anual (de facultad) y realiza la divulgación. Si no es aprobado esto se comunica al departamento correspondiente. La DPG condensa y publica Plan de Ofertas de Postgrado Anual-UCLV en la red y en los OACE.

Como resultado en ese plan aparecen las fechas de los PGA que serán ejecutados en ese período (esto incluye las continuaciones de ediciones abiertas en algún plan anual anterior y que no han concluido).

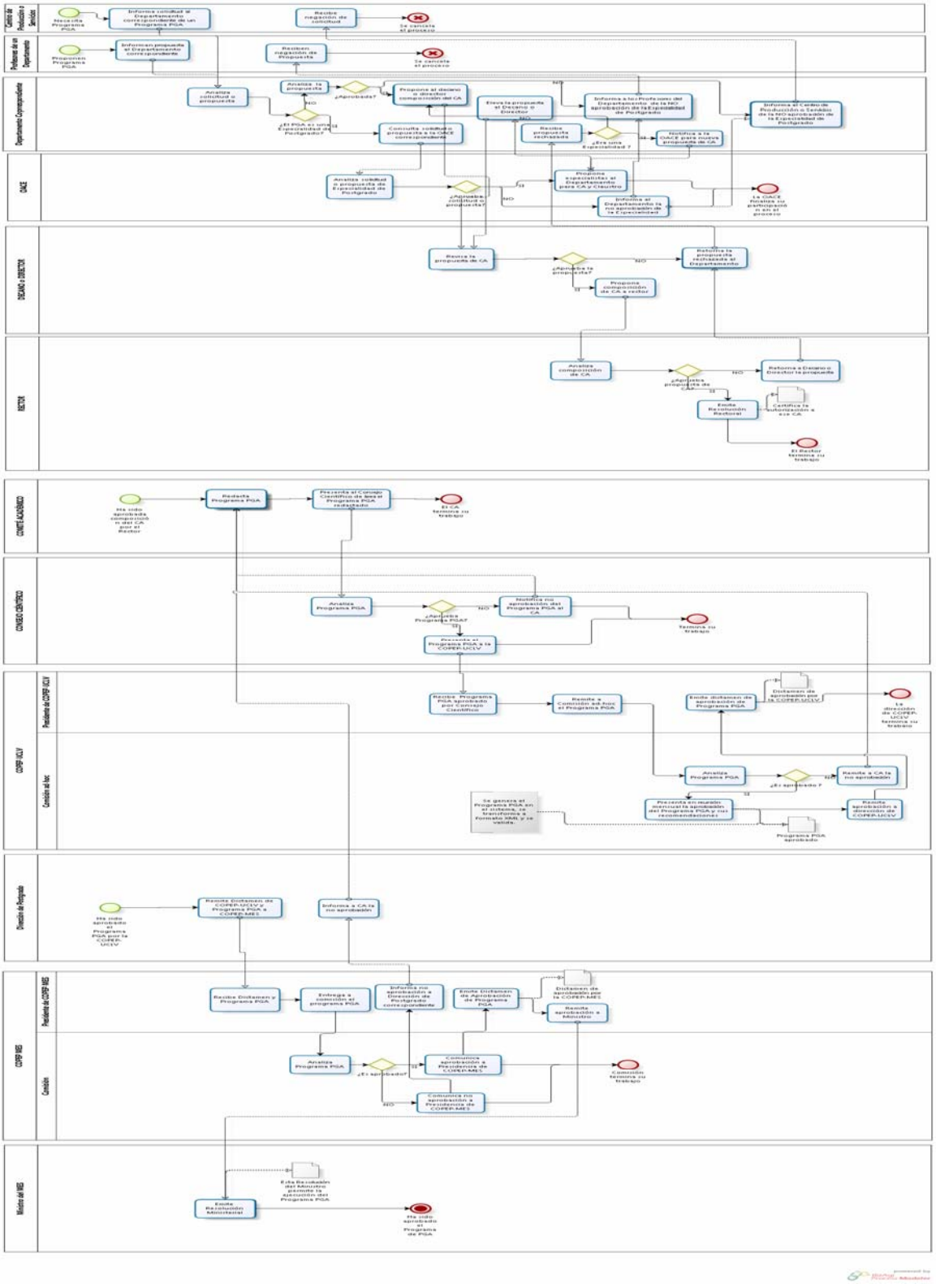

Figura 2.2. Procedimiento de gestación y aprobación de nuevos programas de Maestrías y Especialidades.

Debe recordarse que a los efectos del Plan de Postgrado de un área, los PGA están conformados por distintos tipos de Postgrados de los considerados de Superación Profesional (Cursos, Entrenamientos o Diplomados) y que estos pueden ser matriculados por especialistas interesados de forma independiente a la matrícula en el PGA (Figura 2.3).

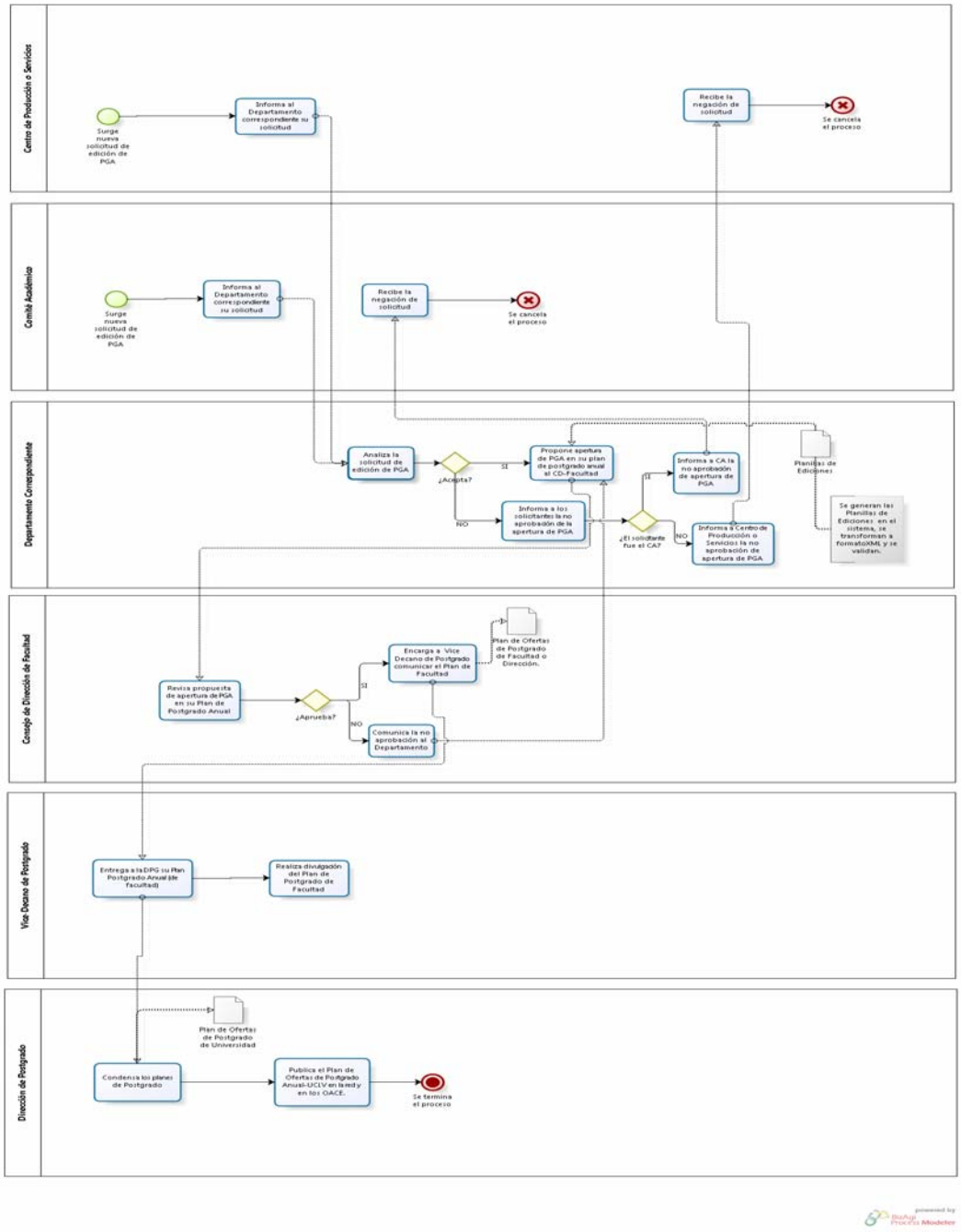

Figura 2.3. Sub-procedimiento de planificación de inicio de Ediciones de Programas de Maestrías y Especialidades.

b. Sub-procedimiento oficial de inicio de Ediciones de Programas de Maestrías y Especialidades.

Tres meses antes de la Fecha Inicial, el CA-PGA solicita la autorización de apertura al Rector con Fechas de Convocatoria, Entrevistas, Matrícula, Inicio y Fin (según programa) de Edición. En diez días a través de la DPG y el asesor jurídico, el Rector firma la Resolución Rectoral de apertura de Edición PGA. La DPG hace llegar copia a la Facultad/Dirección. Si el Rector firma la Resolución Rectoral entonces el CA- PGA lanza una convocatoria dos meses antes de inicio, y comienza a recibir los documentos: Ficha de Matrícula, Fotocopia Legal del Título Universitario, Currículum Vitae firmado en todas sus páginas, Carta de Autorizo de Centro de Trabajo, Foto y demás requerimientos específicos. Nota: La Fotocopia Legal del Título Universitario tiene que ser cotejada contra original por la Secretaria Técnica-Docente de Facultad (persona que tiene firma autorizada según la Resolución Rectoral 42/08) o legalizada en el caso de graduados en el extranjero según explica la Resolución Rectoral 44/08 (PGI) y revisada por la Secretaría General de la UCLV. Si el Rector no firma Resolución Rectoral de apertura de Edición pudiera ser por las siguientes razones: porque se estime no adecuado, porque el PGA debe acreditar antes, porque el PGA debe cerrar o porque la solicitud sea incorrecta. El CA-PGA realiza entrevistas u otras acciones de prematrícula y define una semana antes de inicio oficial el Dictamen Oficial de Matrícula y entrega toda la documentación a la Secretaría de Postgrado de Facultad/Dirección. El CA-PGA entrega copia de Dictamen Oficial de Matrícula a: DPG (siempre), DRI (si hay extranjeros en Dictamen Oficial de Matrícula), y Secretaría General de la UCLV (siempre). El CA-PGA crea el Expediente de la Edición-PGA con: Copia de Resolución Rectoral, original de Dictamen Oficial de Matrícula, Resumen del Programa para la edición, copia de Convocatoria, Calendario de ejecución de edición, y otros Dictámenes necesarios. Si hay extranjeros en la Edición de PGA, al recibir el Dictamen Oficial de Matrícula la DRI entonces confecciona los Convenios-Contratos a cada estudiante y envía copia a Facultad/Dirección correspondiente. El CA-PGA valora conjuntamente con el Jefe de Departamento Docente el claustro real a trabajar en esta edición de PGA, acorde con el claustro oficial del programa y las disponibilidades, e informa de ello a la Secretaría de Postgrado de Facultad/Dirección. La Secretaría de Postgrado de Facultad/Dirección confecciona los Expedientes de Estudiantes para la

Edición de PGA e incluye toda documentación referida, como copia de Resumen de Programa, copia de Dictamen Oficial de Matrícula y otros documentos asociados a un estudiante en particular. La Secretaría de Postgrado de Facultad/Dirección también controla el Expediente de la Edición de PGA que recibe del CA-PGA y confecciona la Lista de la Matrícula para cada profesor actuante por cursos o entrenamientos, con fechas de inicio y fin y hace entrega controlada de estas a cada profesor, quien al final debe devolver con evaluaciones o junto con Acta del Examen con evaluaciones. Si el Sistema Automatizado de Postgrado está instalado, la Secretaría de Postgrado de Facultad/Dirección inscribe la edición del PGA y los estudiantes, y obtiene esas listas y actas automáticamente (Figura 2.4).

3. <u>Procedimiento de ejecución y control en las áreas de Maestrías y Especialidades</u>.

El CA-PGA valora con el Jefe de Departamento Docente al profesor real que impartirá cada Curso o Entrenamiento de Postgrado (acorde a claustro oficial del Programa-PGA y las disponibilidades) e informa a la Secretaría de Postgrado de Facultad/Dirección. La Secretaría de Postgrado de Facultad/Dirección confecciona las Listas de Matrícula para cada profesor actuante por cursos o entrenamientos, con fechas de inicio y fin y hace entrega controlada de estas a cada profesor. Si el Sistema Automatizado de Postgrado está instalado la Secretaría obtiene esas listas del mismo. Cada profesor actuante desarrolla en el tiempo asignado (fechas inicio y fin) su postgrado y al concluir la evaluación en tiempo no mayor de 2 meses posterior a la fecha final, entrega la lista de matrícula con las evaluaciones a la Secretaría de Postgrado para que se proceda a la confección del Informe Final del Postgrado. El profesor actuante puede además de entregar un Acta de Examen si es estilo de su Facultad/Dirección. El CA-PGA atiende y resuelve por Dictamen cualquier solicitud de Convalidación o Examen de Suficiencia, así como de Matrícula por Excepcionalidad. Si decide que si será convalidado debe informar al Profesor actuante a fin de que este ejecute examen o coloque la evaluación CONVALIDADO, según sea el caso. Copias de ese Dictamen debe entregarse a la Secretaría de Postgrado para colocarla en cada Expediente de Estudiante correspondiente. El profesor publica las notas de su postgrado de alguna forma. El estudiante puede solicitar revisión de evaluación al profesor hasta un período controlado de tiempo, si su nota cambia, el profesor nuevamente publica las notas.

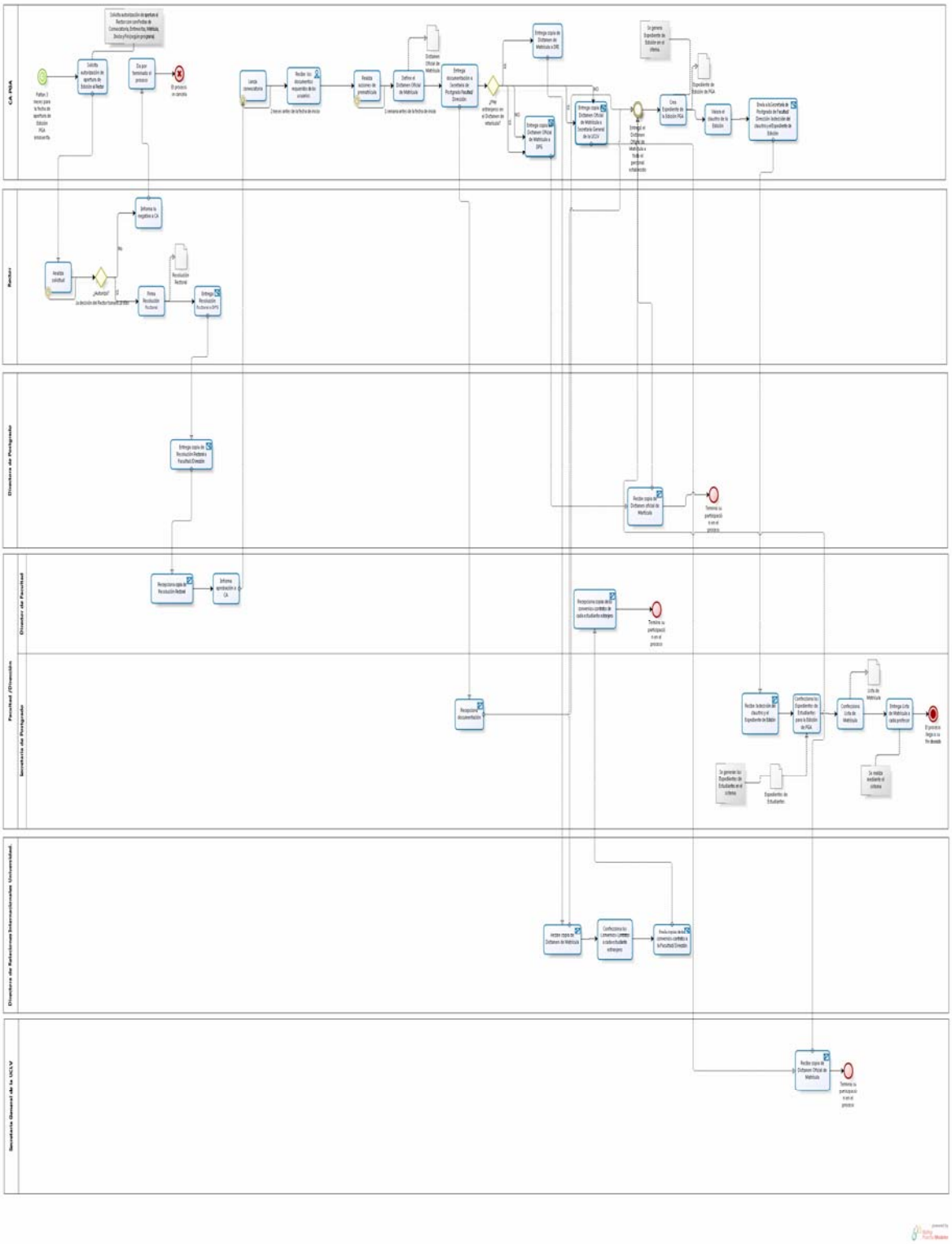

Figura 2.4. Sub-procedimiento oficial de inicio de Ediciones de Programas de Maestrías y Especialidades.

Al mismo tiempo la Secretaría de Postgrado prepara la confección del Informe Final de Postgrado (Curso o Entrenamiento) el cual incluye los resultados docentes de los estudiantes y un resumen del desarrollo del postgrado, cumplimiento, dificultades, etc., y lo pone a revisión y firma del Profesor, Coordinador y VDPG. El CA-PGA controla la ejecución de los cursos o entrenamientos, organizativa y metodológicamente, velando por la entrega a tiempo por cada profesor de la Lista de Matrícula con evaluaciones, o de las Actas de Exámenes, a la Secretaría de Postgrado. El CA-PGA incide en que la Secretaría de Postgrado prepare el Informe Final de Postgrado concluido y que este sea firmado por Profesores, Coordinador y VDPG. Estos documentos van quedando almacenados en el Expediente de la Edición del PGA. El CA-PGA mantiene el control periódico de las notas y los créditos de los estudiantes, además prepara y usa la Sábana de Créditos (lo oferta el Sistema Automatizado). El CA-PGA al mismo tiempo mantiene el control periódico sobre la situación de cada estudiante y procede a emitir Dictamen de Baja si es necesario. Si PGA incluye a estudiantes extranjeros, entonces el CA-PGA debe enviar informe periódico a OCSA (DRI) y el Dictamen Baja si ocurre. El CA-PGA recibe y procesa los documentos probatorios del cumplimiento de actividades que implican créditos investigativos o no lectivos y tomas de decisiones a través de Dictámenes, manteniendo actualizado el Expediente de la Edición del PGA, y entregando copias a la Secretaría de Postgrado para que esta actualice los Expedientes de Estudiantes (Figura 2.5).

Paso 2: Identificar sistemas automatizados.

La lógica de aplicación existente que está, en cualquier extensión, automatizando cualquiera de los requisitos identificados en el Paso 1 necesita ser identificada. Un análisis orientado a servicios no determinará qué tan exactamente los servicios Web encapsularán o reemplazarán la lógica de aplicación legada, eso ayuda a alcanzar los sistemas potenciales afectados.

Los detalles de cómo los servicios Web guardan relación con sistemas existentes son fortalecidos en la fase del diseño orientado a servicios. Por ahora, esta información se usará para ayudar a identificar servicios candidatos de aplicación durante el proceso de modelado de servicios descrito en el Paso 3.

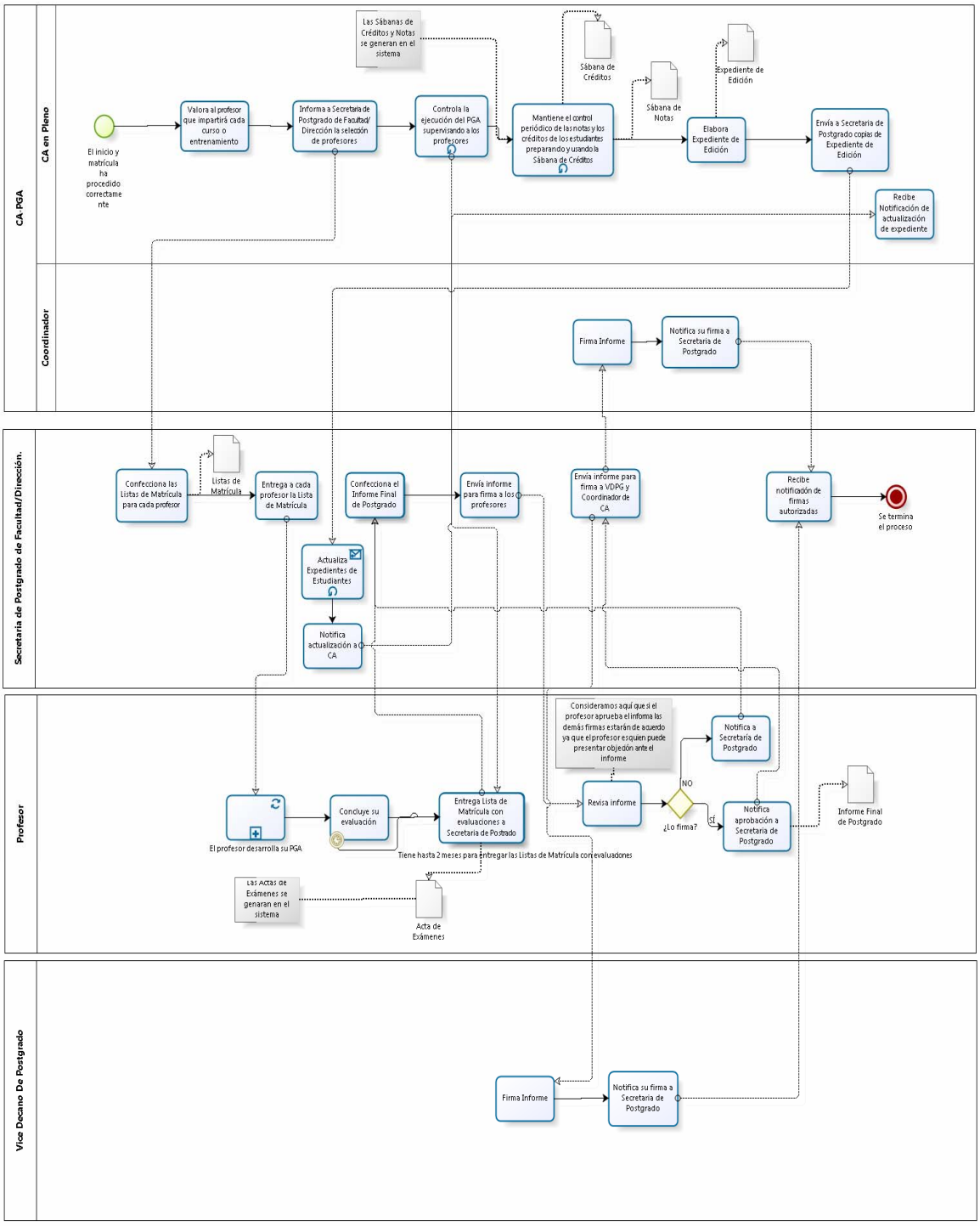

Figura 2.5. Procedimiento de ejecución y control en las áreas de Maestrías y Especialidades.

Este paso esta más perfilado a apoyar los esfuerzos del modelado de soluciones orientadas a servicio de gran escala. Una comprensión de ambientes afectados es todavía útil al modelar una cantidad más pequeña de servicios, pero esta gran cantidad de esfuerzo invertido en investigación no sería requerida en este caso.

Paso 3: Modelar servicios candidatos.

El análisis orientado a servicio introduce el concepto de modelado de servicio, un proceso en el cual las operaciones candidatas de servicios son identificadas y entonces agrupadas en un contexto lógico. Estos grupos eventualmente toman forma como servicios candidatos que son luego ensamblados en un modelo compuesto representando la lógica combinada de la aplicación orientada a servicios planeada.(Erl 2005)

El objetivo principal del análisis orientado a servicios en esta etapa es identificar lo que necesitamos para luego diseñar y construir en las siguientes fases del proyecto. Los pasos a seguir se pueden ver en la Figura 2.6.

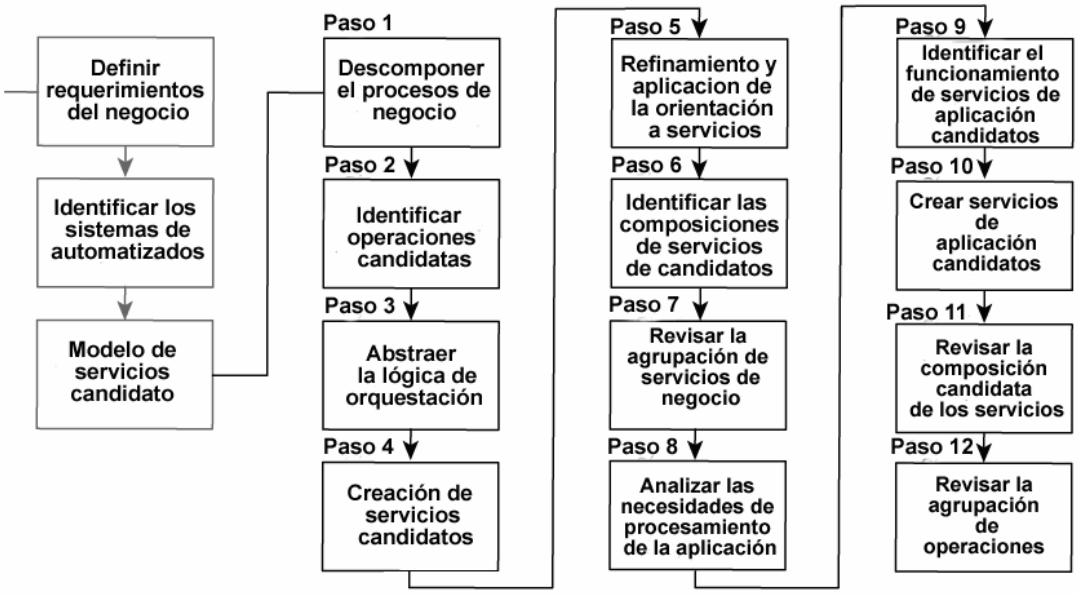

Figura 2.6. Pasos a seguir al modelar los servicios candidatos.

Paso 1: Descomponer el proceso de negocio.

Se toma la documentación del proceso de negocio y se divide en una serie de pasos granulares del proceso. Es importante que la lógica del flujo de trabajo de un proceso esté descompuesta en la representación más granular posible de pasos de procesamiento, que

puede diferir del nivel de granularidad en el cual los pasos de proceso estaban originalmente documentados. En este caso el proceso está dividido en procedimientos bien definidos y de la manera más granular posible, por lo que no se requiere.

Paso 2: Identificar operaciones candidatas.

Algunos pasos dentro de un proceso de negocio pueden ser fácilmente identificados como que no forman parte de la lógica potencial que debe ser encapsulada por un servicio candidato**.** Dentro de estos se incluyen:

- Pasos de procesamiento manual que no pueden ni deben ser automatizados.
- Los pasos del proceso realizados por la lógica existente para la cual la encapsulación del servicio candidato no es una opción.

Entonces procederemos a identificar cuáles son las operaciones candidatas y cuáles deberían considerarse tareas manuales que no requieren automatización.

1. ***Procedimiento de gestación y aprobación de nuevos Programas de Maestrías y Especialidades.***

   - Propuesta del centro de producción o Servicios de Programa de PGA, o propuesta del programa de PGA por profesores del departamento. (Tarea manual)
   - Analizar propuesta el departamento correspondiente. (Tarea manual)
   - Si el PGA=EspPG entonces consultar a OACE su parecer y solicitar especialistas para el CA y el claustro. (Tarea manual)
   - Departamento propone al Decano composición del Comité Académico. (Tarea manual)
   - Si el Decano aprueba la propuesta propone composición del CA al Rector. En caso contrario regresa la propuesta al departamento. (Tarea manual)
   - Rector analiza y aprueba la composición del CA-PGA. En caso de que no lo apruebe retorna la solicitud al Decano. **(Tarea manual)**
   - CA redacta el Programa de PGA. **(Tarea manual)**
   - CA presenta Programa de PGA al Consejo Científico del área correspondiente. **(Tarea manual)**

- Consejo Científico analiza y aprueba. En caso de que no lo apruebe lo retorna al CA. (Tarea manual)
- CA presenta programa a COPEP-UCLV. **(Tarea manual)**
- COPEP-UCLV remite a comisión ad-hoc. **(Tarea manual)**
- La comisión ad-hoc analiza y presenta en comisión mensual de aprobación. En caso de que no lo apruebe retorna la solicitud al CA. **(Tarea manual)**
- COPEP-UCLV emite Dictamen. **(Tarea manual)**
- Dirección de PG remite Dictamen y Programa aprobado por COPEP-UCLV a COPEP-MES. **(Tarea manual)**
- COPEP-MES entrega a la Comisión correspondiente. **(Tarea manual**)
- La Comisión analiza y propone la aprobación definitiva. En caso de que no lo apruebe retorna la solicitud al CA. **(Tarea manual)**
- COPEP-MES emite dictamen de aprobación al Ministro. **(Tarea manual)**
- El Ministro emite Resolución Ministerial que permite la ejecución del programa dado. **(Tarea manual)**
- Se genera en el sistema el Programa de Postgrado Académico. **(Tarea del sistema)**
- Se transforma el Programa de Postgrado Académico a formato XML. **(Tarea del sistema)**
- Se chequea la validez del documento XML. **(Tarea del sistema)**

2. ***Procedimiento de Inicio, Matrícula y Ejecución y Control de Ediciones de Programas de Maestrías y Especialidades existentes.***

   a. ***Sub-procedimiento de planificación de inicio de ediciones de Programas de Maestrías y Especialidades.***

      - Se hace solicitud de Centro de Producción o Servicios de Edición PGA, o surge una nueva propuesta de Edición de PGA por Comité Académico. **(Tarea manual)**
      - Departamento correspondiente analiza la solicitud o propuesta. **(Tarea manual)**
      - Departamento propone apertura de PGA en su Plan de Postgrado Anual. **(Tarea manual)**

- CD-Fac aprueba PlanPG Anual, en caso que no lo apruebe se retorna la propuesta al Departamento. **(Tarea manual)**
- VDPG entrega a DPG su Plan de Postgrado Anual y realiza su divulgación. (Tarea manual)
- DPG condensa y publica Plan de PG Anual en red y OAEC. **(Tarea manual)**
- Se genera Planillas de Ediciones de los PG. **(Tarea del sistema)**
- Se transforman las Planillas de Ediciones a formato XML. **(Tarea del sistema)**
- Se chequea la validez del documento XML. **(Tarea del sistema)**

b. Sub-procedimiento de inicio oficial de ediciones de Programas de Maestrías y Especialidades.

- Se solicita Aut-Apertura al Rector con fechas de convocatoria, entrevistas, matrícula, inicio y fin con tres meses de antelación a la fecha de inicio de CA-PGA. **(Tarea manual)**
- En 10 días a través de DPG y Asesor Jurídico Rector se firma RR de Apertura Edición PGA, se termina el proceso en caso de que se estime no adecuado, porque la solicitud es incorrecta, etc. **(Tarea manual)**
- DPG hace llegar copia a la Facultad/Dirección. **(Tarea manual)**
- CA-PGA lanza convocatoria 2 meses antes del inicio. **(Tarea manual)**
- CA-PGA comienza a recibir los documentos requeridos (Ficha Matrícula, Fotocopia Legal Título Universitario, Curriculum Vitae firmado, Carta Autorizo del CT, etc.). **(Tarea manual)**
- CA-PGA realiza entrevistas u otras actividades prematrícula. **(Tarea manual)**
- CA-PGA define Dictamen Oficial de Matrícula (DOM) una semana antes del inicio. **(Tarea manual)**
- CA-PGA entrega toda la documentación a Sec-PG Facultad/Dirección. **(Tarea manual)**
- CA-PGA entrega copia de DOM a DPG, a DRI si hay extranjeros y a SG. **(Tarea manual)**

- CA-PGA crea Expediente de Edición PGA (requiere copia de RR, DOM original, Resumen de Programa para edición y otros dictámenes necesarios). **(Tarea manual)**
- Se genera Expediente de Edición en el sistema. **(Tarea del sistema)**
- Sec-PG Facultad/Dirección confecciona Expedientes de Estudiantes para Ediciones PGA e incluye toda la documentación requerida. **(Tarea manual)**
- Se genera Expediente de Estudiante en el sistema. **(Tarea del sistema)**
- CA-PGA valora con JDD claustro real a laborar en Ediciones PGA. **(Tarea manual)**
- Sec-PG Facultad/Dirección inscribe PGA y estudiantes en el sistema y se generan las Listas de Matrícula para cada profesor. **(Tarea del sistema)**
- Se genera Actas de Exámenes. **(Tarea del sistema)**

3. ***Procedimiento de ejecución y control en las áreas de Maestrías y Especialidades.***

- CA-PGA valora con JDD claustro real a laborar en Ediciones PGA. **(Tarea manual)**
- Sec-PG Facultad/Dirección inscribe PGA y estudiantes en el sistema y obtiene las Listas de Matrícula que debe entregar a cada profesor. **(Tarea del sistema)**
- Cada profesor desarrolla en el tiempo asignado su postgrado, al concluir la evaluación en tiempo no mayor a dos meses posterior a la fecha de fin, entrega la lista de la matrícula con las evaluaciones a la Sec-PG o de las Actas de Exámenes, esto incide en que Sec-PG prepare el Informe Final de PG concluido y que sea firmado por profesores, Coordinador y VDPG. Estos documentos van quedando almacenados en Expediente de Edición PGA. **(Tarea manual)**
- Se genera Expediente de Edición PGA. **(Tarea del sistema)**
- CA-PGA mantiene control periódico de notas y créditos de estudiantes. **(Tarea del sistema)**
- Se genera Sábana de Créditos. **(Tarea del sistema)**
- Se genera Sábana de Notas. **(Tarea del sistema)**
- CA-PGA mantiene control periódico sobre la situación de los estudiantes y procede a Dictamen de Baja si es necesario. **(Tarea manual)**
- En caso de baja eliminar al estudiante del sistema. **(Tarea del sistema)**

- El CA-PGA recibe y procesa documentos probatorios de cumplimiento de actividades que implican Créditos Investigativos y toma de decisiones a través de Dictámenes, manteniendo actualizado Expediente de Edición y entregar copias a Sec-PG para que actualice Expedientes de Estudiantes. **(Tarea manual)**
- Se actualizan Expediente de Edición y Expedientes de Estudiantes. **(Tarea del sistema)**

Paso 3: Abstraer la lógica de la orquestación.

Cuando se desea construir una capa de orquestación como parte de nuestra SOA, entonces se deben identificar las partes de la lógica de procesamiento que esta capa potencialmente abstrae. (En caso de que no se decida construir una capa de orquestación se omite este paso). Los posibles tipos de lógica para esta capa son:

- Reglas de negocio.
- Lógica condicional.
- Lógica de excepciones.
- Lógica de secuencias.

Estas formas de lógica de orquestación pueden o no ser representadas exactamente por una descripción de pasos. Por ejemplo, algunas descripciones de pasos de procesamiento constan de una condición y una acción (si la condición X ocurre, entonces realice la acción Y). En este caso, sólo remueva la condición y deje la acción.

En este paso se debe tener en cuenta que algunas de estas lógicas de flujo de trabajo identificadas probablemente cambien. Esto se debe a que no todos los pasos del procesamiento necesariamente tengan que ser operaciones de los servicios. En este momento sólo se están creando las operaciones candidatas. Al adentrarse en la fase de diseño, las consideraciones prácticas entran en juego y puede surgir la necesidad de eliminar algunas de las operaciones candidatas, que también requeriría que la lógica de flujo de trabajo correspondiente sea removida de la capa de orquestación. En este trabajo no es necesaria una capa de orquestación por lo que este paso se omite.

Paso 4: Creación de servicios de negocio candidatos.

En este paso se deben revisar los pasos del proceso que quedan y determinar uno o más contextos lógicos con los cuales estos pasos pueden ser agrupados. Cada contexto representa un servicio candidato. Los contextos dependerán de los tipos de servicios de negocio que se han decidido fortalecer.

Por ejemplo los servicios de negocios centrados en tareas requieren un contexto específico en el proceso, mientras que los servicios de negocios centrados en entidades tienen la necesidad de introducir un grupo de pasos de procesamiento de acuerdo a su relación con las entidades antes definidas.

Servicios Candidatos:

1. ***Procedimiento de gestación y aprobación de nuevos Programas de Maestrías y Especialidades.***

- **Programa de Postgrado Académico:** Este servicio es el encargado de generar el Programa de Postgrado Académico y de convertir dicho documento de su formato nativo a XML con el fin de poder enviarlos en el cuerpo de un mensaje SOAP y guardarlos en bases de datos. Esto se conforma en el procedimiento de gestación y aprobación de nuevos Programas de Maestrías y Especialidades, enmarcando el contexto lógico “Programa de Postgrado Académico” donde se realiza las siguientes operaciones:
    - Se genera en el sistema el Programa de Postgrado Académico.
    - Se transforma el Programa de Postgrado Académico a formato XML.
    - Se chequea la validez del documento XML.

2. ***Procedimiento de inicio, matrícula, ejecución y control de Ediciones de Programas de Maestrías y Especialidades existentes.***

a) Sub-procedimiento de planificación de inicio de Ediciones de Programas de Maestrías y Especialidades.

- **Planillas de Ediciones:** Este servicio es el encargado de crear las planillas de ediciones y garantizar su uso posterior transformándolas a formato XML, estas planillas se

conforman en el sub-procedimiento de planificación e inicio de Ediciones de Programas de Maestrías y Especialidades. En este contexto se realizan las siguientes operaciones:

- Se genera Planillas de Ediciones de los PG.
- Se transforman las Planillas de Ediciones a formato XML.
- Se chequea la validez del documento XML.

b) Sub-procedimiento de inicio oficial de Ediciones de Programas de Maestrías y Especialidades.

- **Expediente de Edición:** Este servicio se encarga de crear el expediente de edición que se conforma en el sub-procedimiento de inicio oficial de Ediciones de Programa de Maestría y Especialidades y consta de la siguiente operación:
  - Se genera Expediente de Edición en el sistema.
- **Expediente de Estudiante:** Este servicio se utiliza con el propósito de crear los expedientes de estudiantes, y consta de la siguiente operación:
  - Se genera Expediente de Estudiante en el sistema.
- **Lista de Matrícula:** Este servicio se crea con el fin de buscar las listas de matrícula para cada profesor.
  - Se chequean las listas de matrículas para cada profesor.
- **Actas de Exámenes:** Este servicio se ocupa de generar el acta de exámenes y consta de la siguiente operación:
  - Se generan Actas de Exámenes.

3. Procedimiento de ejecución y control de las aéreas de Maestrías y Especialidades.

- **Sábana de Créditos:** Este servicio se encarga de generar la sábana de créditos.
  - Se genera Sábana de Créditos.
- **Sábana de Notas:** Este servicio se encarga de generar la sábana de notas.
  - Se genera Sábana de Notas.
- **Actualización:** Este servicio es el encargado de actualizar varios documentos, y consta de las siguientes operaciones:
  - Se actualiza Expediente de Edición.

- Se actualiza Expediente de Estudiante.
- En caso de baja eliminar al estudiante del sistema (eliminar de Lista de Matrícula).

En la Figura 2.7 se muestra como quedan los servicios candidatos según sus contextos lógicos.

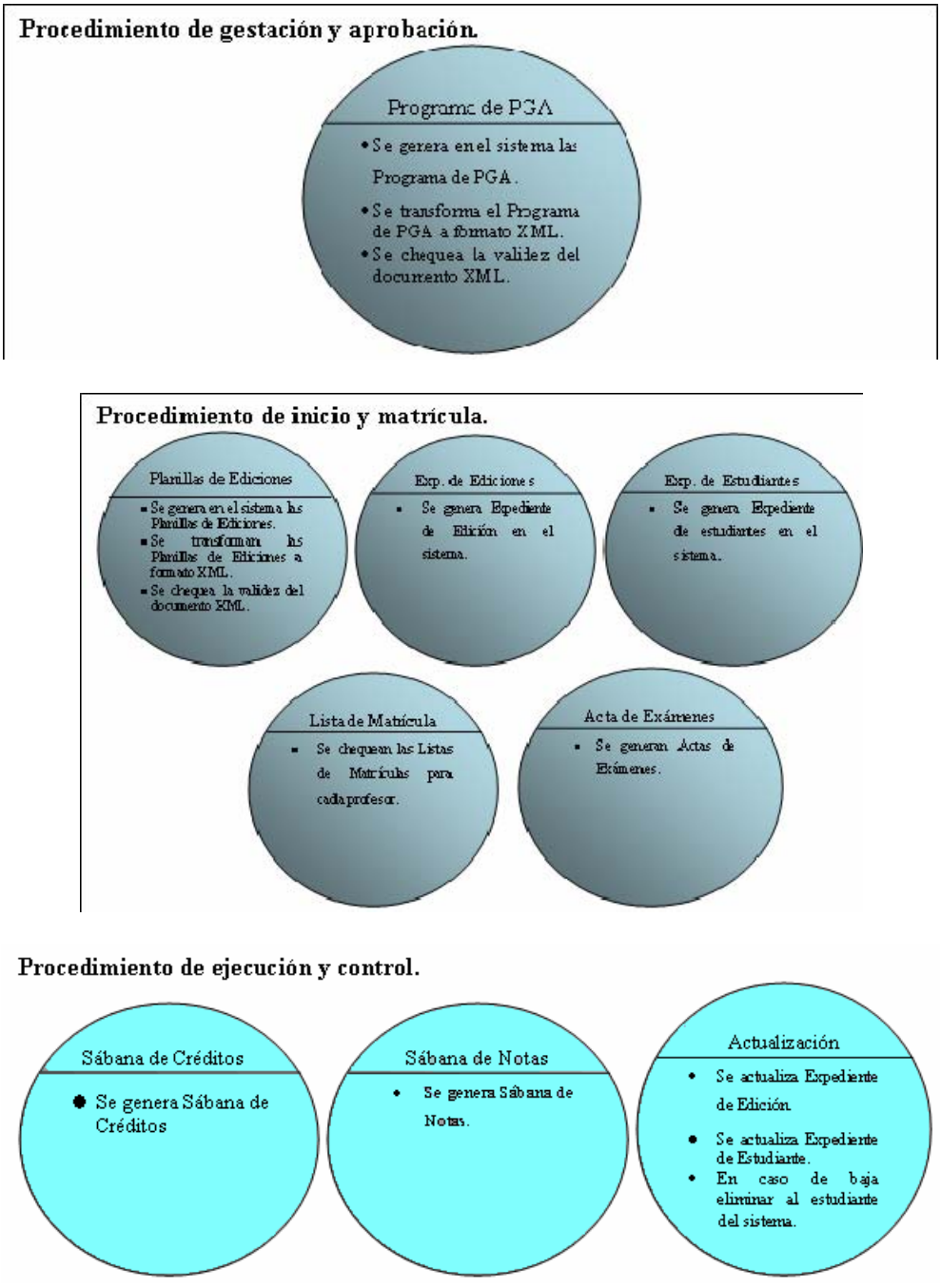

Figura 2.7. Servicios de negocio candidatos.

Paso 5: Refinamiento y aplicación de la orientación a servicios.

Hasta ahora solamente se han agrupado pasos de procesamiento derivados de un proceso de negocio existente. Para hacer los servicios candidatos verdaderamente dignos de una SOA, se debe revisar la lógica de cada operación candidata de servicio propuesta.

Este paso da una oportunidad a hacer ajustes y aplicar principios cruciales de orientación a servicio. A continuación se identifican cuatro principios, los cuales no son inherentemente cumplidos mediante el uso de servicios Web:

- Reutilización
- Autonomía
- Ausencia de estado
- Descubrimiento

De estos cuatro, sólo los primeros dos son importantes para esta etapa en el modelado de servicios. Los dos últimos en esta lista se tratan en el proceso de diseño orientado a servicios. Por lo tanto, este paso se enfoca en lograr que cada operación de los servicios candidatos se identifiquen como potencialmente reutilizables y autónomos.

En este modelo de servicios candidatos se observa la similitud de las operaciones “Se transforma el Programa de Postgrado Académico a formato XML” y “Se chequea la validez del documento XML” del servicio Programa de Postgrado Académico con “Se transforman las Planillas de Ediciones a formato XML” y “Se chequea la validez del documento XML” del servicio Planillas de Ediciones, por lo que se decide crear un nuevo servicio candidato que realice dichas operaciones, logrando así un alto grado de reutilización. Entonces las operaciones se agruparían en el nuevo contexto lógico “Procesamiento de Documento” de la siguiente manera:

- Se transforma el documento a formato XML.
- Se chequea la validez del documento XML.

Luego se unen estas dos operaciones en “Se transforma el documento a XML”. No se menciona la acción de validación debido a que los documentos XML automáticamente son asignados a su esquema correspondiente, y la validación es por consiguiente una parte

inherente del proceso de transformación. Así se logra una mayor autonomía en este servicio candidato, y quedaría conformado este servicio como se muestra en la Figura 2.8.

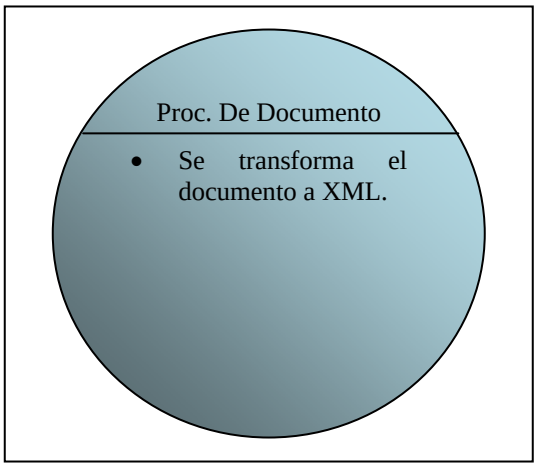

Figura 2.8. Servicio candidato Procesamiento de Documento.

En el servicio Actualización se decide unir las dos operaciones “Se actualiza Expediente de Edición” y “Se actualiza Expediente de Estudiante” en una sola operación llamada “Se actualiza expediente”, haciéndola más genérica, logrando con esto incrementar la reutilización del servicio (Figura 2.9).

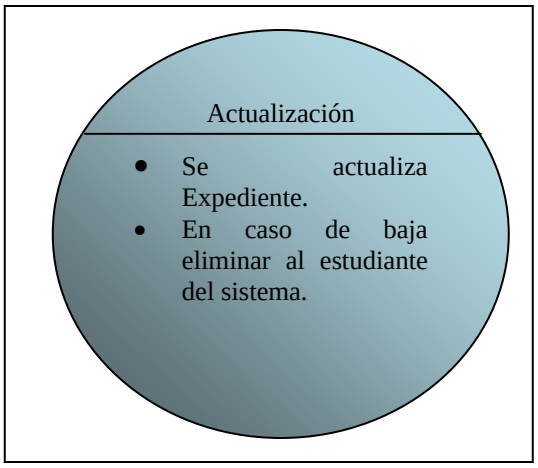

Figura 2.9. Servicio candidato Actualización.

Paso 6: Identificar las composiciones de servicios candidatos.

En este paso se identifica el conjunto de escenarios más comunes que puede tener lugar dentro de los límites del proceso de negocio. Para cada escenario, se siguen los pasos de procesamiento, tal y como se muestra a continuación.

- Tener una buena idea de cómo proceder con la agrupación de las etapas del proceso.
- Poner de manifiesto la posible relación entre la orquestación y la capa de servicios de negocio.
- Identificar el potencial de las composiciones de servicios.
- Poner de relieve la falta de lógica en el de flujo de trabajo o en los pasos del proceso.

No se puede pasar por alto que como parte de los escenarios que se elijan, se incluyan condiciones de falla que involucren la lógica de excepción. También que cualquier capa de servicio que se establezca es preliminar, y todavía sujeta a revisiones durante el proceso de diseño. En este caso se decide utilizar la configuración utilizada en la Figura 2.10. Una propuesta inicial de cómo agrupar las composiciones de servicios candidatos de las distintas etapas del proceso se muestra en la Figura 2.11, donde se aprecia la aparición de nuevos servicios que surgen de la necesidad de lograr una mejor composición, estando ubicado en la subcapa de servicios de negocio centrado en tareas perteneciente a la capa de servicios de negocio. Estos servicios se encargan del control de las operaciones que se relacionan con la lógica de los subprocesos y actúan como servicios controladores de las composiciones de servicios.

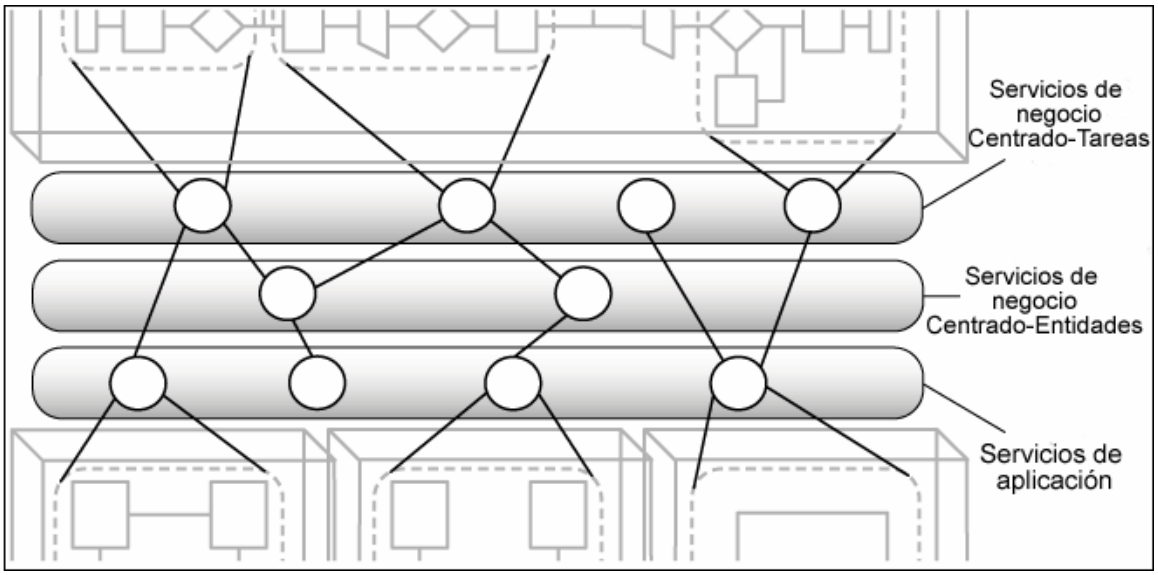

Figura 2.10. Escenario de configuración.

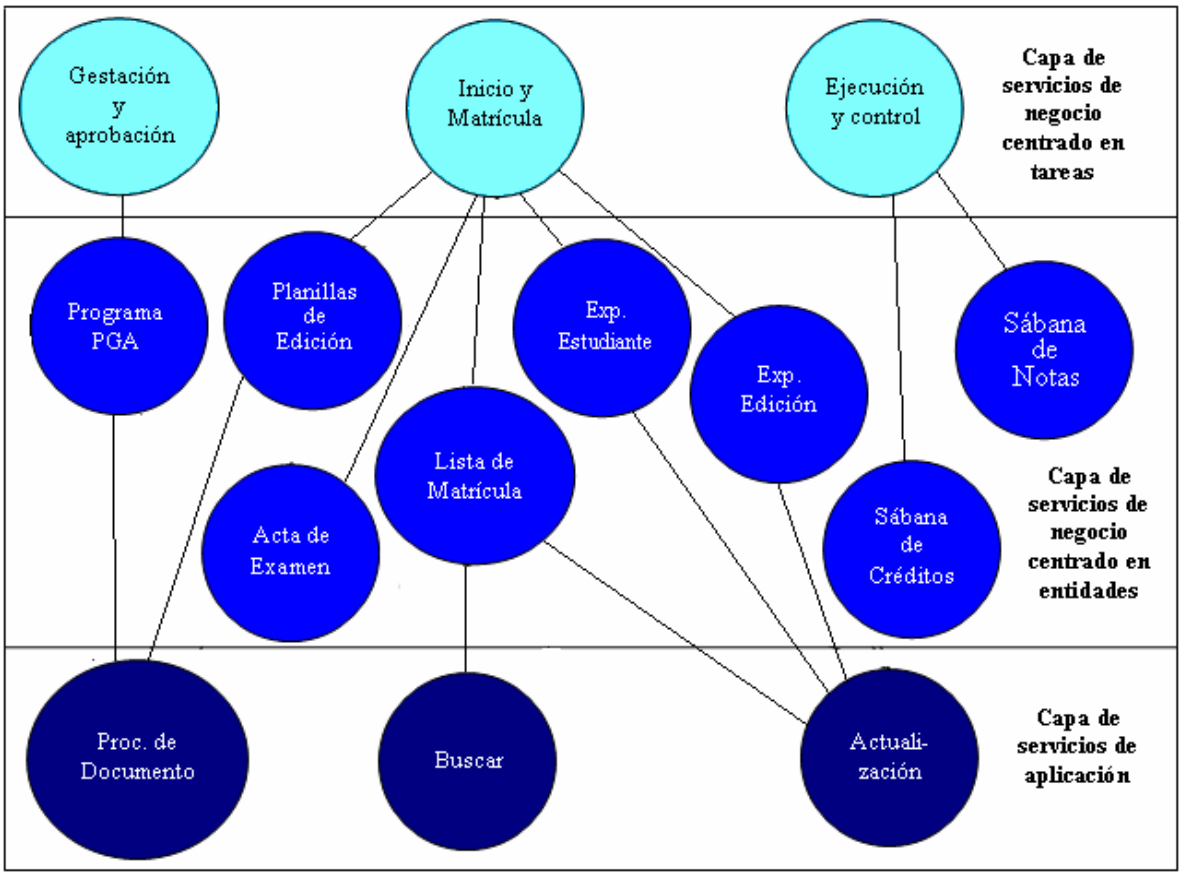

Figura 2.11. Composición del Proceso de Maestrías y Especialidades.

Los servicios más difíciles de identificar son los de aplicación, puesto que ellos deben cumplir con ciertas rigurosidades, para que se tenga un idea lo que se quiere lograr con estos servicios, es que ellos tengan una independencia de los servicios de negocio y de orquestación o controladores, donde sólo serían utilitarios, usados por los servicios de las capas superiores, ahora cabe preguntase que ventaja da esta filosofía, que en caso de que cambie la plataforma sobre la que se encuentran corriendo las aplicaciones, sólo bastaría realizar un grupo de cambios en estos servicios de mas bajo nivel, que son los que interactúan directamente con las aplicaciones, sin tener que cambiar los demás servicios. En esta fase surgen proposiciones iniciales de que servicios pueden ser de aplicación pero como se expone anteriormente no son muy fáciles de identificar y es por ello que pueden surgir nuevos servicios de este tipo. Hasta ahora sólo se han identificado dos de ellos Procesamiento de Documento y Actualización, pero se ha adicionado otro llamado Buscar.

Paso 7: Revisar la agrupación de servicios del negocio.

Basándose en los resultados de las composiciones realizadas en el paso anterior, se revisa el agrupamiento de los pasos del proceso de negocio y las organización de las operaciones de servicio candidatas como sea necesario. No es inusual consolidar o crear grupos nuevos (servicios candidatos) en este punto, pero en este caso las agrupaciones servicio del negocio se mantienen sin variación.

Pasos 8-12.

Los restantes pasos del modelado de servicios candidatos (8-9-10-11-12) no son de obligatorio cumplimiento puesto que sólo se realizan en caso de arquitecturas complejas que necesiten un alto nivel de análisis, donde es necesario hacer revisiones y tener en cuenta la aparición de nuevas agrupaciones con el fin de lograr mejor rendimiento y mayor integración.

2.2. Conclusiones Parciales

En base a los pasos establecidos por la literatura consultada se realizó el análisis orientado a servicios del proceso sustantivo Formación de Másteres y Especialistas para el desarrollo futuro de un Sistema de Control del Postgrado en base a la Arquitectura Orientada a Servicios.

CAPÍTULO 3. DISEÑO ORIENTADO A SERVICIOS DEL SISTEMA DE GESTIÓN DE POSTGRADO.

CAPÍTULO 3: DISEÑO ORIENTADO A SERVICIOS DEL SISTEMA DE GESTIÓN DE POSTGRADO.

El diseño orientado a servicio es el proceso en el cual los diseños concretos de servicios físicos derivan de servicios candidatos lógicos y luego se ensamblan en composiciones abstractas que implementan un proceso de negocio. Las metas globales al realizar un diseño orientado a servicio son las siguientes:

- Determinar las extensiones de la arquitectura.
- Colocar los límites de la arquitectura.
- Identificar las normas requeridas del diseño.
- Definir diseños abstractos de la interfaz de servicio.
- Identificar composiciones de servicio.
- Evaluar el apoyo a los principios de orientación a servicio.
- Explorar el soporte de las características de SOA contemporánea.

Al igual que con el análisis orientado a servicio, primero establecemos un proceso que comienza con algún trabajo preparatorio. Esto conduce a una serie de procesos iterativos que gobiernan la creación de diferentes tipos de diseños de servicio y, finalmente, el diseño del flujo de trabajo global de la solución. De acuerdo con las capas de servicios identificadas anteriormente, la secuencia de diseño que se emplea es la siguiente:

1. Servicios de negocio centrados en entidades.
2. Servicios de aplicación.
3. Servicios de negocio centrados en tareas.

Esta secuencia guía este proceso de diseño, aunque el mismo no sigue siempre una secuencia clara ya que implica volver a revisarlo de manera iterativa e incremental. Es bueno tomar en cuenta lo que la bibliografía señala como prerrequisitos para la fase de diseño orientado a servicios, que por problemas de espacio aparece en el Anexo 1 de este trabajo.

Por otra parte debe destacarse que para el diseño se utilizó la herramienta XMLSpy 2009 sp1.

3.1 Diseño de servicios de negocio centrados en entidades.

Los servicios de negocio centrados en entidades representan la capa de servicio que es menos influenciada por las otras. Su propósito es representar exactamente las entidades de datos correspondientes definidas dentro de los modelos de negocio de una organización. Estos servicios son estrictamente una solución construida para su reutilización por parte de cualquier aplicación que necesite acceder o manejar información asociada con una entidad particular.

A causa de que existen más bien automáticamente en relación con otras capas de servicio, es beneficioso diseñar los servicios de negocio centrados en entidades antes de los otros. Esto establece una capa de servicio abstracta alrededor de la cual el proceso y la lógica de la aplicación subyacente pueden ser situados.

Para el diseño de los servicio de este tipo se deben revisar los servicios existentes para garantizar que no existan funcionalidades traslapadas. También se pueden añadir operaciones a estos servicios en caso de que sea necesario. En nuestro caso contamos con los siguientes servicios de negocio centrados en entidades:

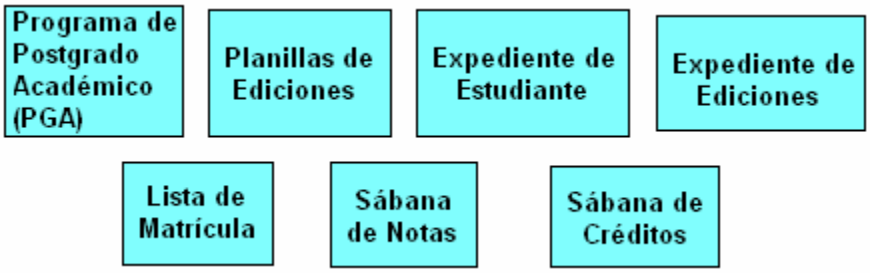

Figura 3.1. Servicios de negocio centrados en entidades.

Servicio Programa de Postgrado Académico:

El servicio Programa de Postgrado Académico perteneciente al proceso de gestación y aprobación tras el refinamiento de los servicios candidatos se quedó con una sola operación “Generar Programa de Postgrado Académico”, ahora se muestra como es que se realiza el diseño de este servicio y como queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de Datos:

Los tipos de datos se definen dentro de la definición *types* del WSDL o en un XSD utilizándolo desde el WSDL mediante la instrucción *import*. Para la operación "Generar Programa de Postgrado Académico" se decide crear el tipo de dato complejo PPGA que contiene los atributos "ID" y "Descripción" como se muestra en la Figura 3.2.

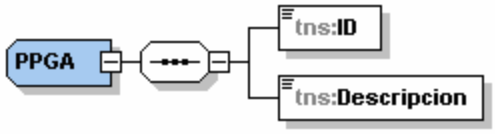

Figura 3.2. Tipo de dato complejo PPGA.

De esta forma queda el código en el WSDL:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="PPGA">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
        <xs:element name="Descripcion" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="PPGA_input">
  <wsdl:part name="parameter" type="tns:PPGA"/>
</wsdl:message>
<wsdl:message name="NewMessageResponse">
  <wsdl:part name="parameter" type="xs:string"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="ProgramaPGA">
  <wsdl:operation name="GenerarPPGA">
    <wsdl:input message="tns:PPGA_input"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="MiBinding" type="tns:ProgramaPGA">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="GenerarPPGA">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
  </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
<wsdl:service name="PPGA">
  <wsdl:port name="Mipuerto" binding="tns:MiBinding">
    <soap:address location="http://localhost/**"/>
  </wsdl:port>
</wsdl:service>
```

Después de mostrar las definiciones que componen la descripción del servicio, se puede ver la Figura 3.3 donde queda conformado de forma general el servicio.

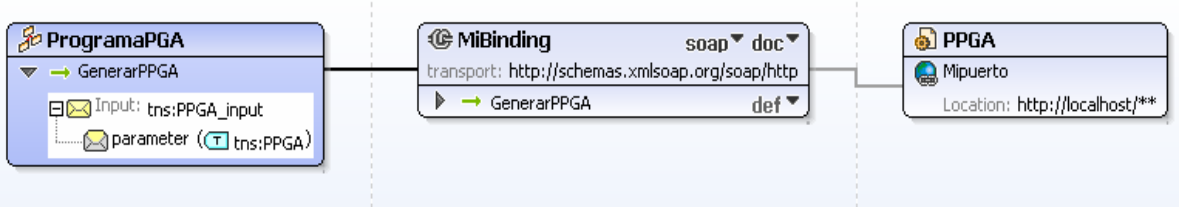

Figura 3.3. WSDL del servicio Programa de Postgrado Académico.

Servicio Planillas de Ediciones:

El servicio Planillas de Ediciones perteneciente al sub-procedimiento de planificación de inicio de ediciones de Programas de Maestrías y Especialidades tras el refinamiento de los servicios candidatos quedó con una sola operación “Generar Planillas de Ediciones”, a continuación se muestra como es que se realiza el diseño de este servicio y como queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de datos:

Para la operación “Generar Planillas de Ediciones” se decide crear el tipo de dato complejo **Plantillas_Ediciones** que contiene el atributo “Descripción”, como se muestra en la Figura 3.4.

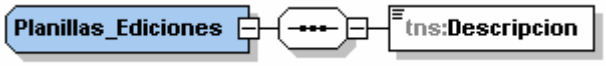

Figura 3.4. Tipo de dato complejo Plantillas_Ediciones.

El código en el WDSL se muestra a continuación:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="Planillas_Ediciones">
      <xs:sequence>
        <xs:element name="Descripcion" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="Planillas_input">
  <wsdl:part name="parameter" type="tns:Planillas_Ediciones"/>
</wsdl:message>
<wsdl:message name="NewMessageResponse">
  <wsdl:part name="parameter" type="xs:string"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="PlanillasEdiciones">
  <wsdl:operation name="GenerarPlanillas">
    <wsdl:input message="tns:Planillas_input"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="MiBinding" type="tns:PlanillasEdiciones">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="GenerarPlanillas">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
  </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
<wsdl:service name="P_Ediciones">
  <wsdl:port name="MiPuerto" binding="tns:MiBinding">
    <soap:address location="http://localhost/**"/>
  </wsdl:port>
</wsdl:service>
```

Después de mostrar las definiciones que componen la descripción del servicio, se puede ver la Figura 3.5 donde queda conformado de forma general el servicio.

Figura 3.5. WSDL del servicio Planillas de Ediciones.

Servicio Expediente de Estudiantes:

El servicio Expediente de Estudiantes del sub-procedimiento de inicio oficial de ediciones de Programa de Maestrías y Especialidades es el encargado de crear este documento en el sistema. En este caso se decide agregar la operación “Actualizar Expediente de Estudiante” que será la encargada de llamar al servicio de aplicación Actualización que se encarga de esta tarea. A continuación se muestra como es que se realiza el diseño de este servicio y como queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de datos:

En la operación “Generar Expediente de Estudiantes” se crea el tipo de dato complejo Exp_Estudiante que consta de los atributos “ID” y “Descripción”, ya que esta operación es la encargada de generar en el sistema el documento Expediente de Estudiantes (Figura 3.6).

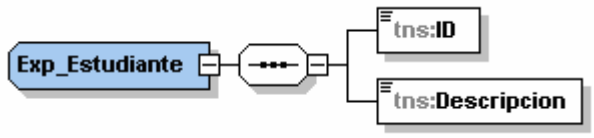

Figura 3.6. Tipo de dato complejo Exp_Estudiante.

En la operación “Actualizar Expediente de Estudiante” se decide crear el tipo de dato complejo ExpToUpdate que contiene el atributo “ID” y “Update” (Figura 3.7), lo cual está encaminado a identificar qué expediente se quiere actualizar y qué se quiere actualizar de él. También se utiliza el tipo de dato anteriormente comentado Exp_Estudiante.

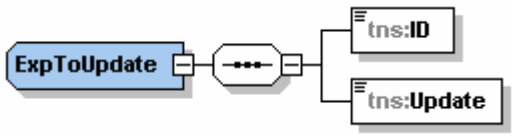

Figura 3.7. Tipo de dato complejo ExpToUpdate.

El código en el WSDL se muestra a continuación:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="Exp_Estudiante">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
        <xs:element name="Descripcion" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ExpToUpdate">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
        <xs:element name="Update" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="ExpEstudiante_input">
  <wsdl:part name="parameter" type="tns:Exp_Estudiante"/>
</wsdl:message>
<wsdl:message name="NewMessageResponse">
  <wsdl:part name="parameter" type="xs:string"/>
</wsdl:message>
<wsdl:message name="UpdateFileMessageRequest">
  <wsdl:part name="parameter" type="tns:ExpToUpdate"/>
</wsdl:message>
<wsdl:message name="UpdateFileMessageResponse">
  <wsdl:part name="parameter" type="tns:Exp_Estudiante"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="ExpedienteEstudiante">
  <wsdl:operation name="GenerarExpEstudiante">
    <wsdl:input message="tns:ExpEstudiante_input"/>
  </wsdl:operation>
  <wsdl:operation name="Actualizar_Expediente">
    <wsdl:input message="tns:UpdateFileMessageRequest"/>
    <wsdl:output message="tns:UpdateFileMessageResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="Mi_Binding" type="tns:ExpedienteEstudiante">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="GenerarExpEstudiante">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
  </wsdl:operation>
  <wsdl:operation name="Actualizar_Expediente">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
<wsdl:service name="ExpEstudiante">
  <wsdl:port name="MiPuerto" binding="tns:Mi_Binding">
    <soap:address location="http://localhost/**"/>
  </wsdl:port>
</wsdl:service>
```

Después de mostrar las definiciones que componen la descripción del servicio, se puede ver la Figura 3.8 donde queda conformado de forma general el servicio.

Figura 3.8. WSDL del servicio Expediente de Estudiantes.

Los demás servicios de negocio centrados en entidades se detallan en el anexo 2.

3.2 Diseño de servicios de aplicación.

Los servicios de aplicación son los caballos de trabajo de SOA. Representan la subcapa del fondo de la capa compuesta de servicio, responsable de llevar a cabo cualquiera de las demandas de procesamiento dictadas a ellos por el negocio y la capa de orquestación.(Erl 2005)

A diferencia de los servicios en las capas centradas en negocios, el diseño de servicios de aplicación no precisa experiencia de análisis de negocio. La capa de servicios de aplicación es una abstracción pura, orientada a servicio de los ambientes técnicos de una organización, mejor definida por esos que entienden más de estos ambientes.

Por las muchas consideraciones del mundo real y de tecnologías específicas que necesitan ser tenidas en cuenta, los servicios de aplicación pueden ser el tipo de servicio más difícil para diseñar. Más allá, el contexto establecido por estos servicios puede ser constantemente desafiado, como la tecnología es mejorada o reemplazada, y como la lógica de la aplicación relacionada es construida o alterada.

En este caso se cuenta con los servicios Actualizar y Buscar como se muestran en la figura 3.9.

Figura 3.9. Servicios de aplicación.

Servicio Actualizar:

Este servicio es el encargado de actualizar varios documentos que genera el sistema. A continuación se muestra como es que se realiza el diseño de este servicio y como queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de datos:

Para la operación “Actualizar Expediente” se decide crear el tipo de dato FileToUpdate que contiene “ID” y “Update” como elementos (Figura 3.10).

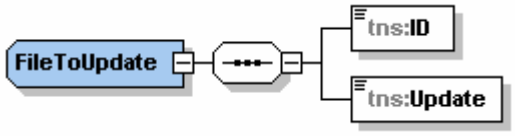

Figura 3.10. Tipo de dato complejo FileToUpdate.

En la operación “En caso de baja se elimina estudiante del sistema” se crean dos tipos de datos complejos, Estudiante_Baja que contiene el atributo “ID” y ListaActualizada que contiene “ID” e “Inf_Lista“(Figura 3.11).

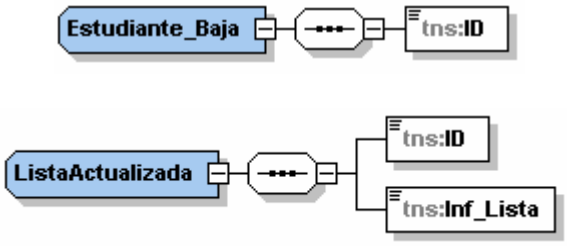

Figura 3.11. Tipos de datos complejos Estudiante_Baja y ListaActualizada.

A continuación se muestra el código en el WSDL:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="FileToUpdate">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
        <xs:element name="Update" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Estudiante_Baja">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ListaActualizada">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
        <xs:element name="Inf_Lista" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="UpdateFile">
  <wsdl:part name="parameter" type="tns:FileToUpdate"/>
</wsdl:message>
<wsdl:message name="UpdateFileMessageResponse">
  <wsdl:part name="parameter" type="xs:string"/>
</wsdl:message>
<wsdl:message name="DarBajaMessagerequest">
  <wsdl:part name="parameter" type="tns:Estudiante_Baja"/>
</wsdl:message>
<wsdl:message name="DarBajaMesasgeResponse">
  <wsdl:part name="parameter" type="tns:ListaActualizada"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="Actualizacion">
  <wsdl:operation name="ActualizarExpediente">
    <wsdl:input message="tns:UpdateFile"/>
  </wsdl:operation>
  <wsdl:operation name="DarBaja">
    <wsdl:input message="tns:DarBajaMessagerequest"/>
    <wsdl:output message="tns:DarBajaMesasgeResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="Mi_Binding" type="tns:Actualizacion">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="ActualizarExpediente">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
  </wsdl:operation>
  <wsdl:operation name="DarBaja">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
<wsdl:service name="Serv_Update">
  <wsdl:port name="MiPuerto" binding="tns:Mi_Binding">
    <soap:address location="http://localhost/**"/>
  </wsdl:port>
</wsdl:service>
```

Después de mostrar las definiciones que componen la descripción del servicio, se puede ver la Figura 3.12 donde queda conformado de forma general el servicio.

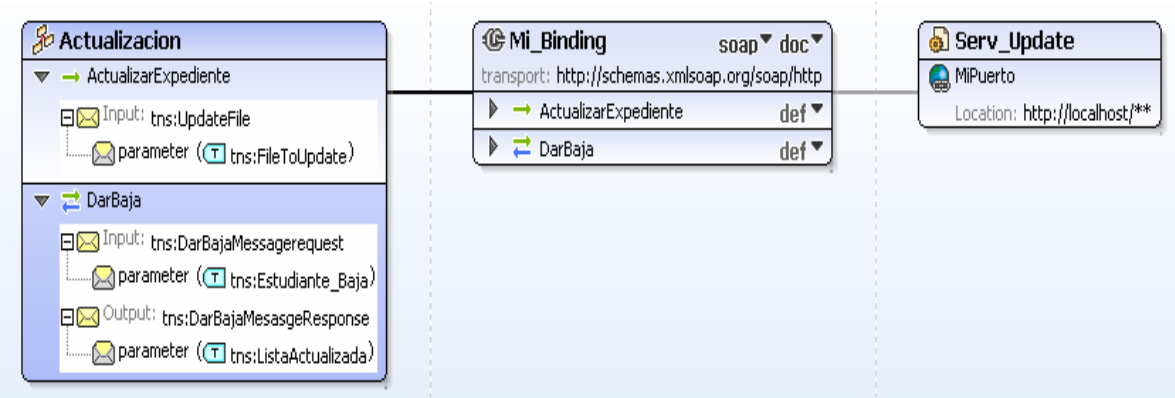

Figura 3.12. WSDL del servicio Actualización.

Servicio Buscar:

El servicio Buscar es el encargado de buscar las listas de matrículas dado el ID de un profesor. A continuación se muestra como es que se realiza el diseño de este servicio y como queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de datos:

Para la operación "Buscar listas de matrícula" se crean dos tipos de datos complejos, Request_info que contiene el elemento "ID" y Response_info que tiene "ID" y "LMatricula_info" (Figura 3.13).

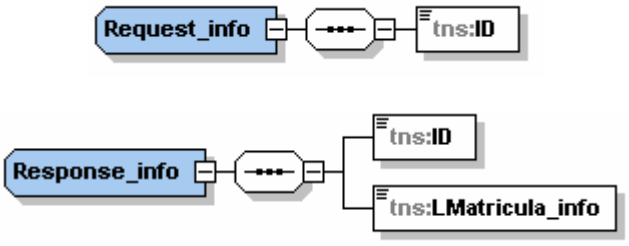

Figura 3.13. Tipos de datos complejos Request_info y Response_info.

A continuación se muestra el código en el WSDL:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="Request_info">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Response_info">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
        <xs:element name="LMatricula_info" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="InfoRequest">
  <wsdl:part name="parameter" type="tns:Request_info"/>
</wsdl:message>
<wsdl:message name="InfoResponse">
  <wsdl:part name="parameter" type="tns:Response_info"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="Buscar">
  <wsdl:operation name="BuscarInfo">
    <wsdl:input message="tns:InfoRequest"/>
    <wsdl:output message="tns:InfoResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="Mi_Binding" type="tns:Buscar">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="BuscarInfo">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
<wsdl:service name="ServicioBuscar">
  <wsdl:port name="MiPuerto" binding="tns:Mi_Binding">
    <soap:address location="http://localhost/**"/>
  </wsdl:port>
</wsdl:service>
```

Después de mostrar las definiciones que componen la descripción del servicio, se puede ver la Figura 3.14 donde queda conformado de forma general el servicio.

Figura 3.14. WSDL del servicio Buscar.

Servicio Procesamiento de Documento:

El servicio Procesamiento de Documento es el encargado de transformar un documento a formato XML para su posterior uso en otras actividades. A continuación se muestra cómo es que se realiza el diseño de este servicio y cómo queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de datos:

Para la operación “Transformar documento a XML” se definen dos tipos de datos complejos TrasformToXMLType que contiene los elementos “SourcePath” y “DestinationPath”, y TransformToXMLReturnCodeType que contiene “Code” y “Message” (Figura 3.15). A continuación se muestra el código en el WSDL:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="TransformToXMLType">
      <xs:sequence>
        <xs:element name="SourcePath" type="xs:string"/>
        <xs:element name="DestinationPath" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="TransformToXMLReturnCodeType">
      <xs:sequence>
        <xs:element name="Code" type="xs:string"/>
        <xs:element name="Message" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

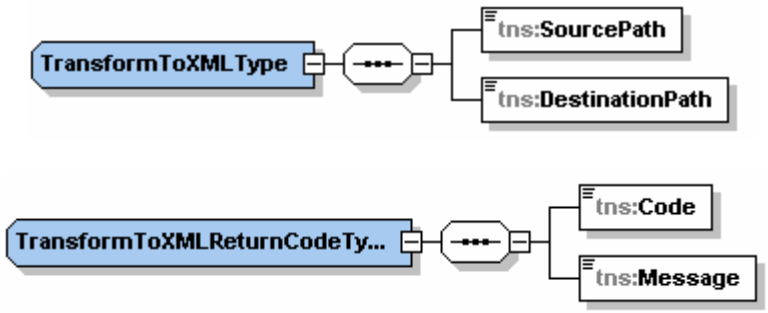

Figura 3.15. TrasformToXMLType y TransformToXMLReturnCodeType.

Mensajes:

```
<wsdl:message name="transformToXMLRequestMessage">
  <wsdl:part name="parameter" type="tns:TransformToXMLType"/>
</wsdl:message>
<wsdl:message name="transformToXMLResponseMessage">
  <wsdl:part name="parameter" type="tns:TransformToXMLReturnCodeType"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="ProcesamientoDocumento">
  <wsdl:operation name="TransformToXML">
    <wsdl:input message="tns:transformToXMLRequestMessage"/>
    <wsdl:output message="tns:transformToXMLResponseMessage"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="Mi_Binding" type="tns:ProcesamientoDocumento">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="TransformToXML">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
<wsdl:service name="Proc_Doc">
  <wsdl:port name="MiPuerto" binding="tns:Mi_Binding">
    <soap:address location="http://localhost/**"/>
  </wsdl:port>
</wsdl:service>
```

Después de mostrar las definiciones que componen la descripción del servicio, se puede ver la Figura 3.16 donde queda conformado de forma general el servicio.

Figura 3.16. WSDL del servicio Buscar.

3.3 Diseño de servicios de negocio centrados en tareas.

El proceso para diseñar los servicios centrados en tareas usualmente precisa menos esfuerzo que los dos procesos previos del diseño, simplemente porque la reutilización no es generalmente una consideración primaria. Por consiguiente, sólo las operaciones de servicio candidatas identificadas como parte del proceso de modelado de servicios son diseñadas aquí. En este caso se cuenta con los siguientes servicios centrados en tareas: Figura 3.17.

Figura 3.17. Servicios de negocio centrados en tareas.

A continuación se realiza el proceso mediante el cual se definen los servicios centrados en tareas, comenzando por el análisis de los servicios con los que interactúa, después la secuencia de las operaciones y finalmente se define la operación que se incluirá en el WSDL del servicio.

Servicio Gestación y Aprobación.

El servicio Gestación y Aprobación se descompone en dos servicios, Programa de Postgrado Académico y Procesamiento de Documento (Figura 3.18), con dos operaciones que se ejecutan en un orden definido en el diagrama de secuencia de la Figura 3.19.

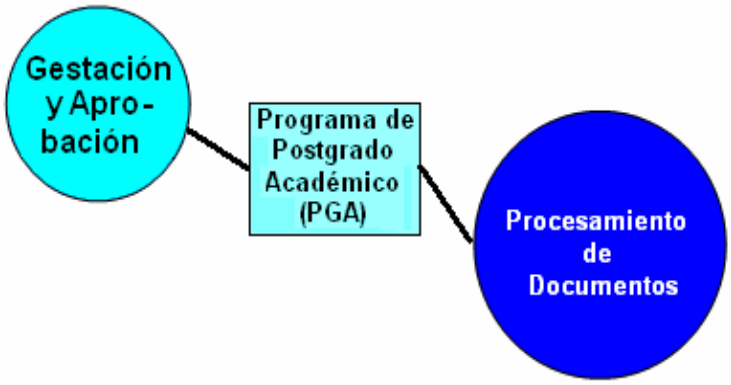

Figura 3.18. Relación entre Gestación y Aprobación con Programa de Postgrado Académico y Procesamiento de Documento.

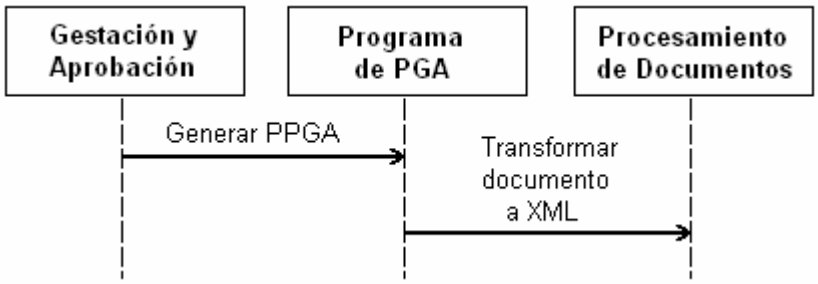

Figura 3.19. Diagrama de secuencia.

Según este diagrama el servicio tendría las siguientes interacciones, Figura 3.20.

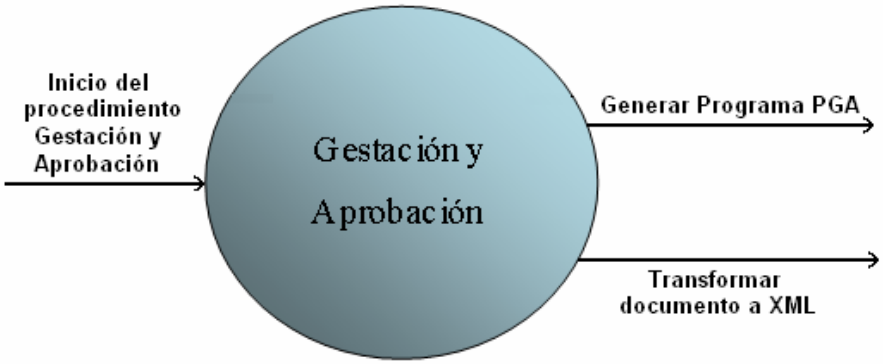

Figura 3.20. Peticiones y respuestas identificadas en el servicio Gestación y Aprobación.

Estas acciones requieren que el servicio actúe como pedidor para iniciar los intercambios de mensajes con otros servicios. Por lo tanto la operación "Inicio del procedimiento de Gestación y Aprobación" es inicializada por el servicio Programa de Postgrado Académico. Esta acción por consiguiente necesita ser expresada en la interfaz del servicio. El nombramiento de la operación es considerado para algo más apropiado como se muestra en la Figura 3.21.

Figura 3.21. Interfaz del servicio Gestación y Aprobación.

Servicio Inicio y Matrícula.

El servicio Inicio y Matrícula se descompone en varios servicios como muestra la Figura 3.22, compuesto por varias operaciones que se ejecutan en un orden definido en el diagrama de secuencia que se muestra en la figura 3.23.

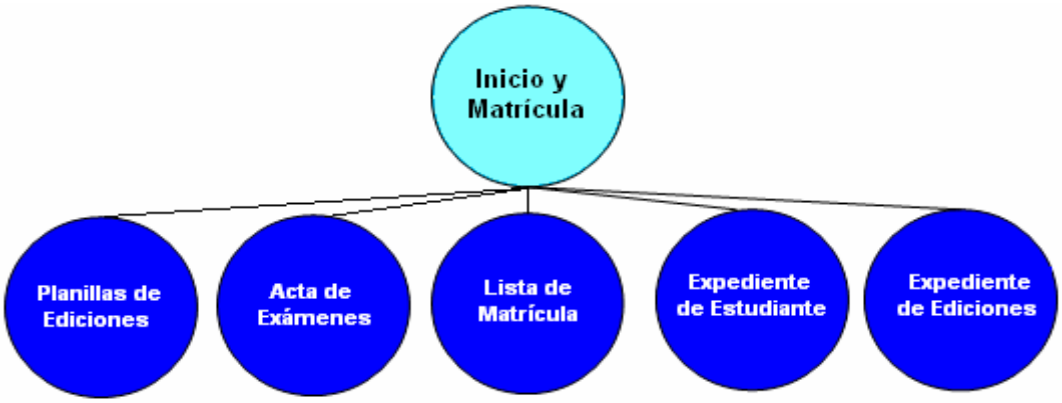

Figura 3.22. Relación del servicio Inicio y Matrícula con Plantillas de Ediciones, Acta de Exámenes, Lista de Matrícula, Expediente de Estudiante y Expediente de Ediciones.

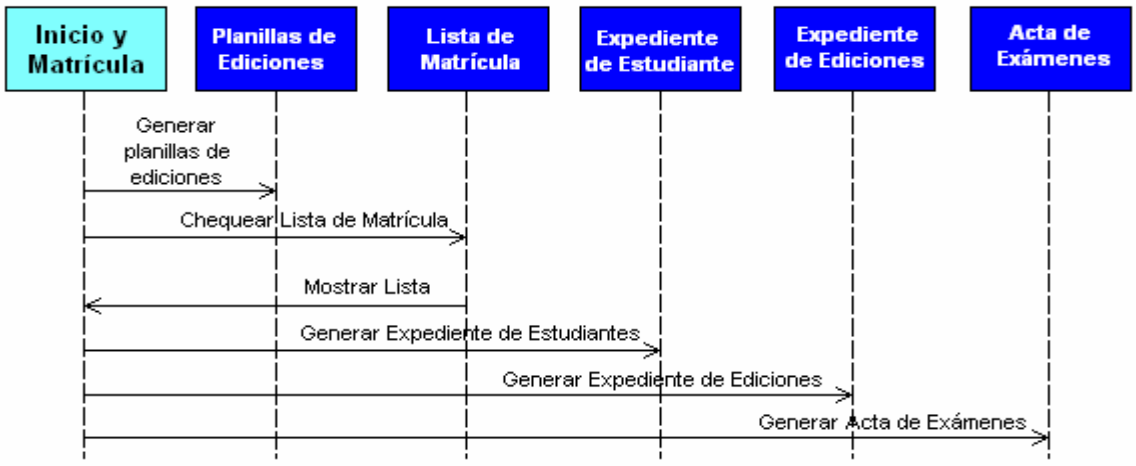

Figura 3.23. Diagrama de secuencia.

Según el diagrama anterior tendría las siguientes interacciones (Figura 3.24).

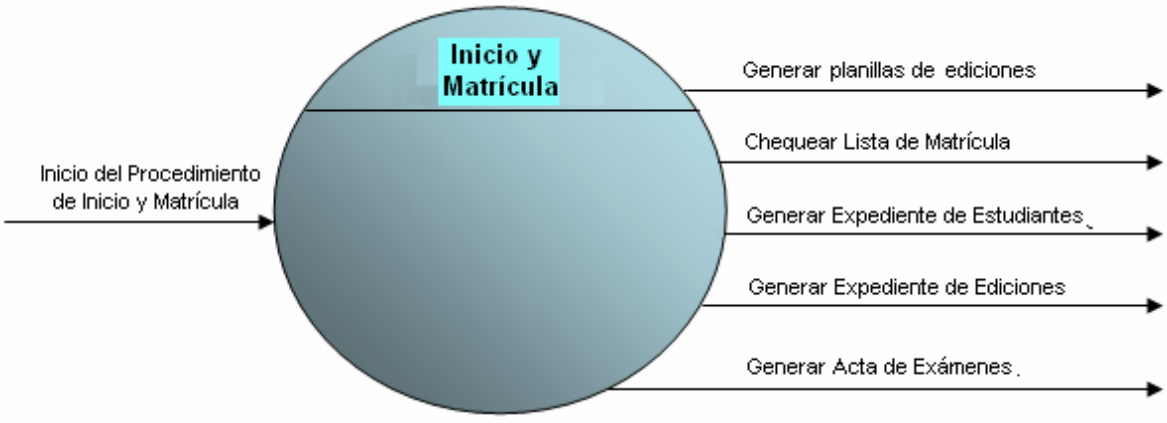

Figura 3.24. Peticiones y respuestas identificadas en el servicio Inicio y Matrícula.

Estas acciones requieren que el servicio actúe como solicitante para iniciar los intercambios de mensajes con otros servicios. Por lo tanto la operación “Inicio del procedimiento de Inicio y Matrícula” es inicializada por el servicio Plantillas de Ediciones. Esta acción por consiguiente necesita ser expresada en la interfaz del servicio.

El nombramiento de la operación es considerado para algo más apropiado como se muestra en la Figura 3.25.

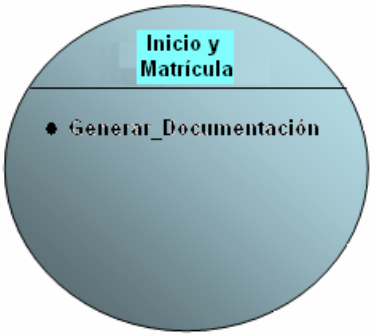

Figura 3.25. Interfaz del servicio Inicio y Matrícula.

Servicio Ejecución y Control.

El servicio Ejecución y Control se descompone en dos servicios como muestra la Figura 3.26, compuesto por varias operaciones que se ejecutan en un orden definido en el diagrama de secuencia que se muestra en la figura 3.27.

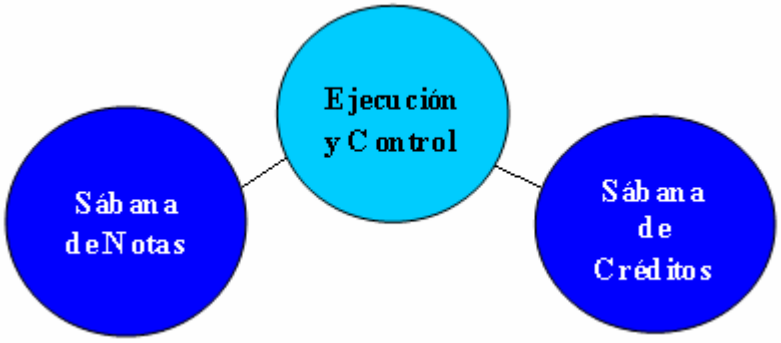

Figura 3.26. Relación del servicio *Ejecución y Control* con *Sábana de Notas* y S*ábana de Créditos*.

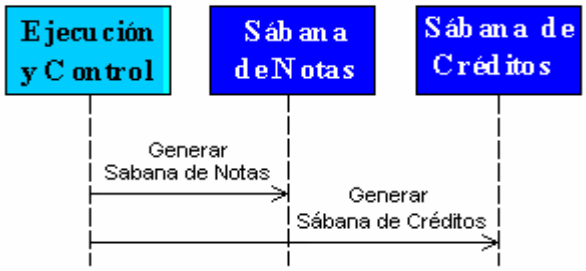

Figura 3.27. Diagrama de secuencia.

Según el diagrama de secuencia anterior tendría las siguientes interacciones (Figura 3.28).

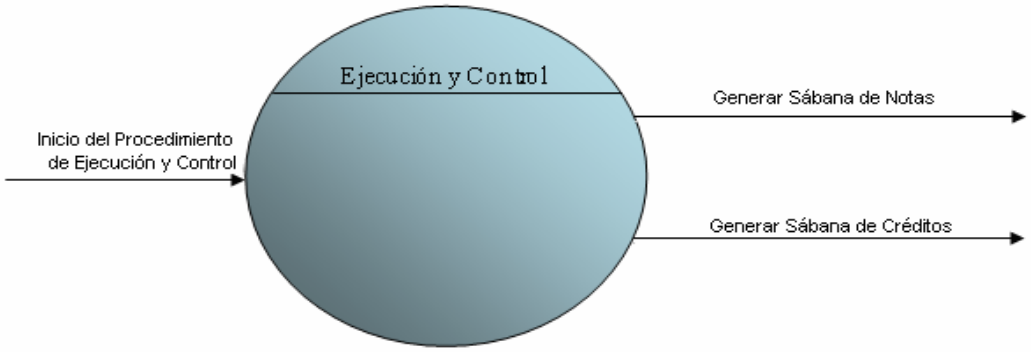

Figura 3.28. Peticiones y respuestas identificadas en el servicio Ejecución y Control.

Estas acciones requieren que el servicio actúe como solicitante para iniciar los intercambios de mensajes con otros servicios. Por lo tanto la operación “Inicio del procedimiento de Ejecución y Control” es inicializada por el servicio Sábana de Notas. Esta acción por consiguiente necesita ser expresada en la interfaz del servicio.

El nombramiento de la operación es considerado para algo más apropiado como se muestra en la Figura 3.29.

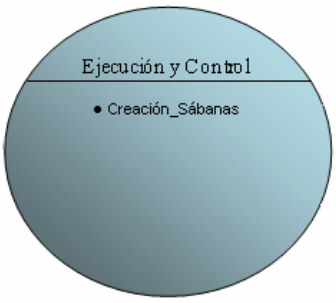

Figura 3.29. Interfaz del servicio Ejecución y Control.

3.4. Conclusiones Parciales

En este capítulo se logran definir y diseñar los servicios de negocio centrados en entidades “Programa de Postgrado Académico”, “Planillas de Ediciones”, “Expediente de Estudiante”, “Expediente de ediciones”, “Lista de Matrícula”, “Sábana de Notas” y “Sábana de créditos”. Se logran definir y diseñar además los servicios de aplicación “Actualizar”, “Buscar” y “Procesamiento de Documentos”, así como también los servicios de negocio centrados en tarea “Gestación y Aprobación”, “Inicio y Matrícula” y “Ejecución y Control”. De ellos se determina su definición abstracta y concreta, y las formas de relacionarse entre sí.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. Se analizaron las actividades relacionadas con los procedimientos del Proceso Sustantivo de Formación de Masteres y Especialidades, determinándose como servicios candidatos “Programa de PGA” en el procedimiento de gestación y aprobación; “Planillas de Ediciones”, “Expediente de Ediciones”, “Expediente de Estudiantes”, “Lista de Matrícula” y “Acta de Examen” en el procedimiento de inicio y matrícula; y también “Actualización”, “Sábana de Créditos” y “Sábana de Notas” en el procedimiento de ejecución y control.
2. Se modelan los servicios candidatos obteniéndose nuevos contextos lógicos en que se agrupan las operaciones correspondientes, dentro de las capas de abstracción que responden a la lógica del proceso de negocio objeto de estudio. La Capa de Servicios de Negocios Centrada en Tareas incluye los servicios “Gestación y Aprobación”, “Inicio y Matrícula” y “Ejecución y Control”, que actúan como servicios controladores. En la Capa de Servicios de Negocios Centrada en Entidades caen los servicios “Programa de PGA”, “Planillas de Ediciones”, “Expediente de Ediciones”, “Expediente de Estudiantes”, “Lista de Matrícula”, “Acta de Examen”, “Sábana de Créditos” y “Sábana de Notas”. En la Capa de Servicios de Aplicación quedan los servicios “Actualización”, “Buscar” y “Procesamiento de Documentos”.
3. Se diseñaron las definiciones WSDL de los servicios de la arquitectura SOA propuesta utilizando la herramienta XMLSpy 2009 sp1.

RECOMENDACIONES

1. Implementar los servicios diseñados en este trabajo para que sean utilizados en un futuro Sistema de Control de Postgrado.

2. Ampliar el diseño con los servicios correspondientes a los otros procesos sustantivos de postgrado, teniendo en cuenta la reutilización de los aquí desarrollados.

REFERENCIAS BIBLIOGRÁFICAS

REFERENCIAS BIBLIOGRÁFICAS

- Anónimo (1995). The Common Object Request Broker Architecture.
- Bell, M. (2008). SERVICE-ORIENTED MODELING. SERVICE ANALYSIS, DESIGN, AND ARCHITECTURE, John Wiley & Sons.
- Box, D., J. d. Vadoss, et al. (2007). Service Oriented Architecture (SOA) in the Real World.
- Chase, N. (2006). Understanding Web services specifications, Part 3: Uniform Description and Discovery Interface (UDDI).
- Chase, N. (2006). Understanding Web Services specifications, Part 2: Web Service Description Language (WSDL)
- Erl, T. (2005). Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall PTR.
- Fensel, D., H. Lausen, et al. (2007). Enabling Semantic Web Services, Springer.
- Gil, J. M. (2002). .Net Remoting.
- Hall, B. (2001). Beej's Guide to Network Programing.
- Herrera, Y. R. (2003) Sistemas Distribuidos.
- Krafzig, D., K. Banke, et al. (2005). Enterprise SOA: Service-Oriented Architecture Best Practices.
- Marshal, D. (1999). Remote Procedure Call (RPC).
- Microsoft (2006) La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real.
- Moreno, R. (2006). Sistema para el Control de la Actividad de Postgrado.
- Mycrosystems, S. (2010) An Overview of RMI Applications.
- Raj, G. S. (1998) A Detailed Comparison of CORBA, DCOM and Java/RMI.
- Szimansky, W. (2003) Extensible .NET remoting framework.
- Zhang, L.-J. (2007). Modern Technologies in Web Services Research, IGI Golbal.
- Zukowsky, J. (2003). Programación Java 2 J2SE 1.4

BIBLIOGRAFÍA

BIBLIOGRAFÍA

- Anónimo (1995) *The Common Object Request Broker Architecture*.
- Baresi, L. and E.D. Nitto (2007) *Test and Analysis of Web Services*.
- Bell, M. (2008) *Service-oriented Modeling*.
- Bell, M. (2010) *SOA Modeling Patterns for Service-Oriented Discovery and Analysis.*
- Box, D., J. d. Vadoss (2007) *Service Oriented Architecture (SOA) in the Real World*.
- Chase, N. (2006) *Understanding Web Services specifications, Part 1: SOAP*.
- Chase, N. (2006) *Understanding Web Services specifications, Part 2: Web Service Description Language (WSDL).*
- Chase, N. (2006) *Understanding Web services specifications, Part 3: Uniform Description and Discovery Interface (UDDI).*
- Chase, N. (2006) *Understanding Web services specifications, Part 4: WS-Security*.
- Chase, N. (2007) *Understanding Web services specifications, Part 5: WS-Policy*.
- Davies, J., et al. (2008) *The Definitive Guide to SOA Oracle Service Bus*.
- Education, P. (2003) *Real World XML Web Services*.
- Erl, T. (2005) *Service-Oriented Architecture: Concepts, Technology, and Design*.
- Fensel, D., et al. (2007) *Enabling Semantic Web Services*.
- Gil, J. M. (2002) .*Net Remoting*.
- Hall, B. (2001) *Beej's Guide to Network Programing*.
- Hasan, J. y M. Duran (2006) *Expert Service-Oriented Architecture in C# 2005*.
- Herrera, Y. R. (2003) *Sistemas Distribuidos*.
- Hurwitz, J., et al. (2007) *Service Oriented Architecture for Dummies*.
- Jennings, F. y D. Salter (2008) *Building SOA-Based Composite Applications Using NetBeans IDE 6*.
- Knutson, J. y H. Kreger (2003) *Web Services for J2EE, Version 1.1*.
- Krafzig, D., K. Banke, y D. Slama (2005) *Enterprise SOA: Service-Oriented Architecture Best Practices*.

- Mandal, M. y H. Silberman (2007) *Understanding Web services specifications, Part 6: WS-Interoperability*.
- Mandal, M. and H. Silberman (2007) *Understanding Web services specifications, Part 7: WS-Business Process Execution Language*.
- Marshal, D. (1999) *Remote Procedure Call (RPC).*
- Microsoft (2006) *La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real*.
- Moreno, R. (2006) *Sistema para el Control de la Actividad de Posgrado*.
- Moreno, R. (2010) *Procedimientos del Proceso Sustantivo de Formación de Másteres y Especialistas de Postgrado. Universidad Central "Marta Abreu" de Las Villas*.
- Mycrosystems, S. (2003) *Java Remote Method Invocation*.
- Mycrosystems, S. (2010) *An Overview of RMI Applications*.
- Raj, G. S. (1998) *A Detailed Comparison of CORBA, DCOM and Java/RMI*.
- Rosen, M., et al. (2008) *Applied SOA: Service-Oriented Architecture and Design Strategies*.
- Rosenberg, J. y D.L. Remy (2004) *Securing Web Services with WS-Security*.
- Ryman, A. (2005) *Developing Web Services with Eclipse*.
- Szimansky, W. (2003) *Extensible .NET remoting framework.*
- Zhang, L.-J. (2007) *Modern Technologies in Web Services Research*.
- Zukowsky, J. (2003) *Programación Java 2 J2SE 1.4*.

ANEXOS

ANEXO 1. PRERREQUISITOS PARA LA FASE DE DISEÑO ORIENTADO A SERVICIOS.

Antes de que se profundice en los detalles del proceso de diseño orientado a servicio, se debería asegurar que se tiene una suficiente comprensión de las partes principales de los lenguajes requeridos para diseñar los servicios. En capítulos anteriores se ha hecho referencia a algunos conceptos y características de la definición WSDL y el protocolo de comunicación SOAP, que en conjunto con *XML Schema Definition Language*, juegan un papel importante en el diseño de los servicios. En la Figura 1 se aprecia como estas tres especificaciones se relacionan entre sí.

Figura 1. Relación entre especificaciones.

Relación de WSDL con *XML Schema Definition Language*.

El *XML Schema Definition Language* (XSD) se ha convertido en una parte central y muy común de XML y arquitecturas basadas en servicios Web. La estructura jerárquica de documentos XML puede estar formalmente definida creando un esquema XSD, por lo tanto un documento XML se considera una instancia de su esquema correspondiente. La estructura establecida dentro de un esquema XSD (Figura 2) contiene una serie de reglas y restricciones para las cuales las instancias del documento XML deben acceder para que los analizadores y procesadores los estimen válido.

Las reglas fundamentales de representación de datos que proveen los esquemas XSD están relacionadas con la representación de datos según el tipo. Al igual que con los tipos de datos usados en los lenguajes de programación, los esquemas XSD proveen un conjunto de tipos de datos no-propietarios utilizados para representar información en las instancias del documento XML. Los esquemas XSD pueden existir como documentos separados

(típicamente identificados por extensiones de archivo .xsd), o pueden estar adjuntados en otros tipos de documentos, como instancias del documento XML o definiciones WSDL. Las instancias del documento XML comúnmente se conectan a un archivo separado de esquema XSD a fin de que el mismo esquema pueda usarse para validar múltiples instancias del documento. Las definiciones WSDL pueden importar los contenidos de un archivo XSD, o también pueden tener contenido de un esquema adjunto en línea.

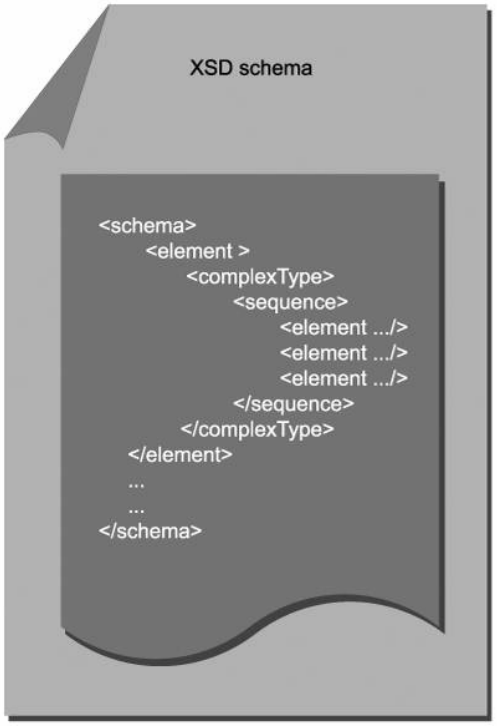

Figura 2. Estructura de un XSD.

Conceptos básicos del lenguaje WSDL.

El Lenguaje de Descripción de Servicios Web es el estándar de las tecnologías fundamentales asociadas con el diseño de los servicios, como se ve a continuación los WSDL son una parte central de todos los aspectos de diseño de servicios donde se establece la idea de las definiciones abstractas y concretas.

Sólo para resumir, la definición abstracta contiene una serie de partes que incluyen los tipos, el mensaje y el tipo de puerto (o interfaz), mientras que la definición concreta se compone de los *binding* y servicios (Figura 3).

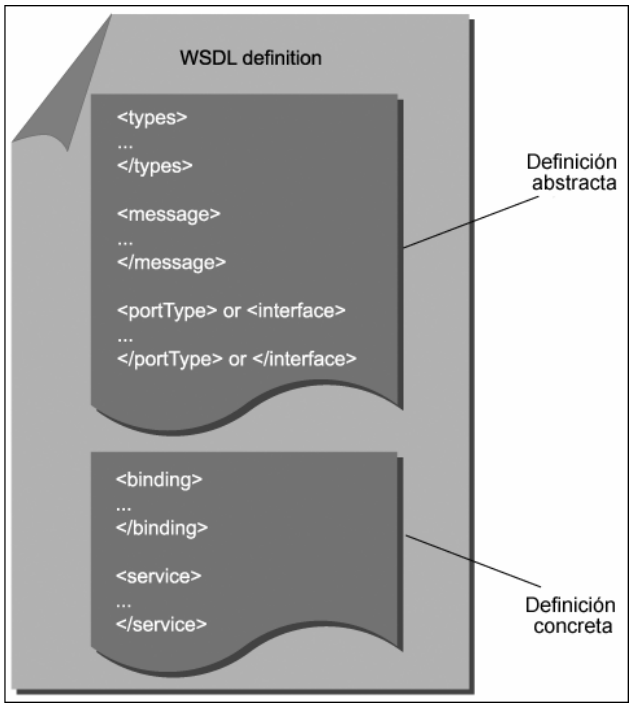

Figura 3 Estructura de un WSDL.

Los WSDL describen la interfaz pública de los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en la definición abstracta y se enlazan después al protocolo concreto de red y al formato del mensaje.

Así, WSDL se usa a menudo en combinación con SOAP y *XML Schema*. Un programa cliente que se conecta a un servicio Web puede leer el WSDL para determinar qué funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de *XML Schema*. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

Partes de un WSDL.

Elemento *types*: Esta etiqueta define las estructuras de datos que se utilizarán para construir los mensajes de petición como de respuesta. Estas estructuras de datos pueden construirse con cualquier lenguaje, pero lo más normal es hacerlo con *XML Schema*. Este apartado es el más complicado sobre todo cuando se tenga que construir estructuras de datos muy complejas.

Elemento *message*: Proporciona una abstracción común para el paso de mensajes entre el cliente y el servidor. Como puede utilizar múltiples formatos de definición de esquema en documento WSDL es necesario disponer de un mecanismo común de identificar los mensajes. El elemento *Message* proporciona este nivel común de abstracción al que se hará referencia en otras partes del documento WSDL. Normalmente aparecen múltiples elementos *Message* en un documento WSDL, uno para cada mensaje que se comunica entre el cliente y el servidor.

Elemento *portType*: Contiene un conjunto de operaciones abstractas que representan los tipos de correspondencia que pueden producirse entre el cliente y el servidor. Para los Servicios Web de estilo RPC se pude pensar en un *portType* como una definición interna en donde cada método se pude definir como una operación. Un tipo puerto se compone de un conjunto de elementos *operation* que define una determinada acción. Los electos *operation* se componen de mensajes definidos en el documento WSDL.

WSDL define cuatro tipos de operaciones:

1. ***Request-response*** (petición-respuesta) comunicación del tipo RPC en la que le cliente realiza una petición y el servidor envía la correspondiente respuesta.
2. ***One-way*** (un-sentido) Comunicación del estilo documento en la que el cliente envía un mensaje pero no recibe una respuesta del servidor indicando el resultado del mensaje procesado.
3. ***Solicit-response*** (solicitud-respuesta) La contraria a la operación petición-respuesta. El servidor envía una petición y el cliente le envía de vuelta una respuesta.
4. ***Notification*** (notificación) La contraria a la operación un-sentido el servidor envía una comunicación del estilo documento al cliente.

Elemento *binding*: Contiene las definiciones de la asociación de un protocolo como SOAP a un determinado *bindingType*. Las definiciones *binding* especifican detalles de formatos del mensaje y el protocolo. Por ejemplo, la información de asociación especifica si se puede acceder a una instancia de un *portType* de forma RPC. Las definiciones *binding* también indican el número de comunicaciones de red que se requieren para realizar una determinada acción.

Elemento *service*: Este elemento indica donde se encuentra el Servicio usando una etiqueta. Cada etiqueta define el formato de los mensajes, y la dirección donde se encuentra el servicio que acepta mensajes en ese formato. Un servicio es un grupo de puertos relacionados y se definen en el elemento service. Un puerto es un extremo concreto de un Servicio Web al que se hace referencia por una dirección única. Los puertos que se definen en determinado servicio son independientes. Por ejemplo, la salida de un puerto que no puede utilizarse como una entrada de otro.

Protocolo de comunicación SOAP.

La estructura de los mensajes SOAP es relativamente simple. Constan de tres secciones, un *header* (encabezado), un *body* (cuerpo), y *fault* (defecto), todas dentro del *envelope* (sobre). Apropiadamente, los elementos que se describen en este epígrafe son representados por los mismos nombres (Figura 4).

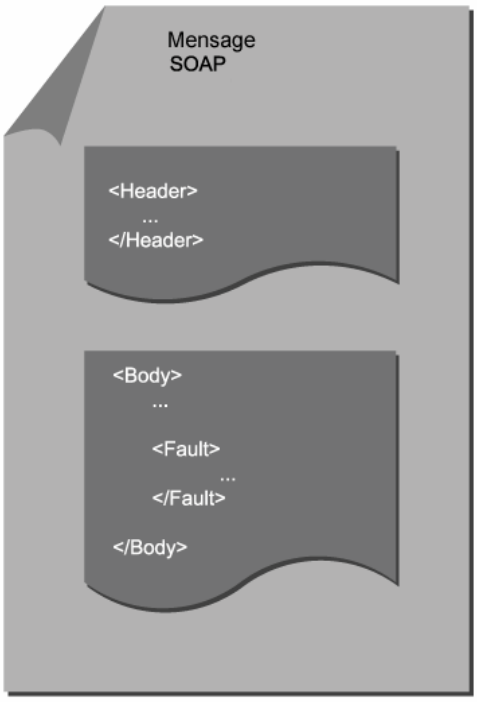

Figura 4. Estructura de un mensaje SOAP.

Elemento *Envelope*: El elemento *Envelope* representa la raíz de las estructuras del mensaje de SOAP. Contiene una construcción obligatoria *Body* y una construcción optativa *Header*.

Elemento *Header*: La sección del encabezado del mensaje SOAP posibilita el conjunto de características de las especificaciones WS-*. La mayor parte de estas extensiones son implementadas en un nivel de mensaje e introducen nuevos bloques de encabezados de SOAP estandarizados destinados a estar adjuntadas en la construcción *Header*.

Elemento *Body*: Éste es el único elemento hijo requerido de la construcción del Envelope de SOAP. Contiene el mensaje con la carga útil con formato XML bien estructurado. La estructura y el nombramiento usado para definir esta parte del mensaje SOAP guardan relación con los atributos *style* y *use*.

Las construcciones *Body* del mensaje SOAP están definidas dentro de las construcciones de mensaje WSDL que, como se ha establecido, hacen referencia a la información del tipo de dato de *XSD Schema* desde la construcción *types* del WSDL (Figura 5). Mientras los bloques de encabezado de SOAP pueden ser tramitados activamente durante la transmisión de un mensaje SOAP, el *Body* no debería ser tocado. Sin embargo, si se permite, servicios intermediarios todavía podrían leer y derivar información del contenido del *Body*.

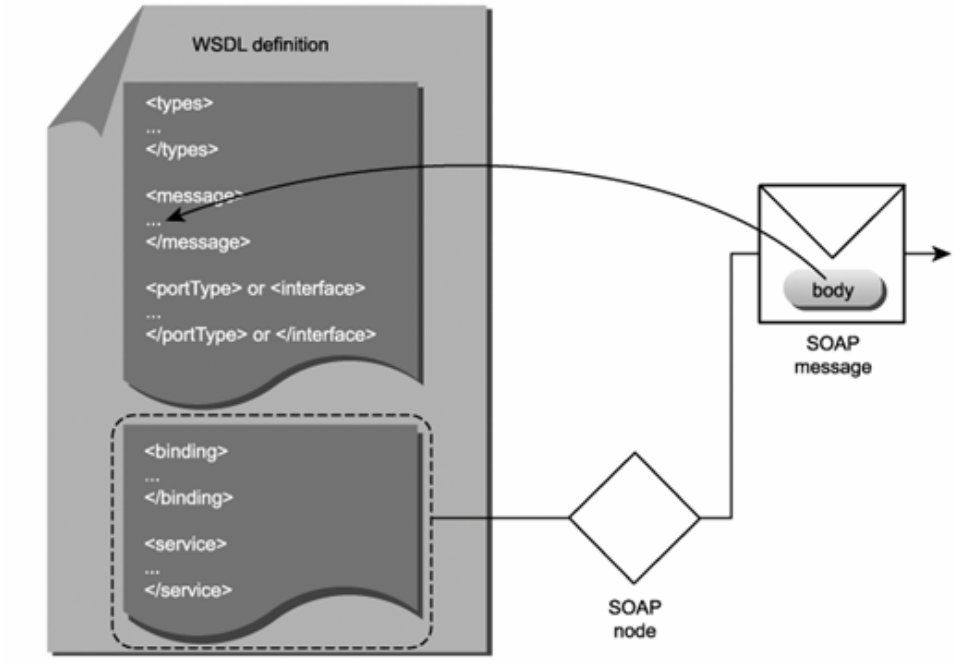

Figura 5. Un *Body* de un mensaje SOAP definido dentro del elemento *message* de un WSDL.

ANEXO 2. OTROS SERVICIOS DE NEGOCIO CENTRADOS EN ENTIDADES.

Servicio Expediente de Ediciones:

Este servicio del sub-procedimiento de inicio oficial de ediciones de Programa de Maestrías y Especialidades es el encargado de crear este documento en el sistema. En este caso se decide agregar la operación “Actualizar Expediente de Ediciones” que será la encargada de llamar al servicio de aplicación Actualización que se encargará de esta tarea. A continuación se muestra como es que se realiza el diseño de este servicio y como queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de datos:

En la operación “Generar Expediente de Ediciones”, se crea el tipo de dato complejo ExpEdiciones que consta de los atributos “ID” y “Descripción”, ya que esta operación es la encargada de generar en el sistema el documento Expediente de Estudiantes (Figura 1).

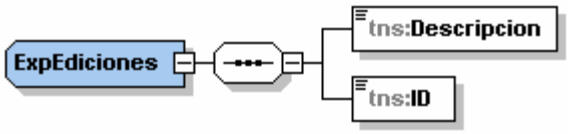

Figura 1. Tipo de dato complejo ExpEdiciones.

En la operación “Actualizar Expediente de Ediciones” se decide crear el tipo de dato complejo EEdicToUpdate que contiene el atributo “ID” y “Update” (Figura 2), lo cual está encaminado a identificar qué expediente se quiere actualizar y qué se quiere actualizar de él. También se utiliza el tipo de dato anteriormente comentado ExpEstudiantes en la operación *response*.

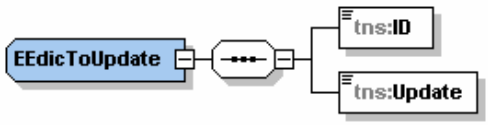

Figura 2. Tipo de dato complejo EEdicToUpdate.

El código en el WSDL se muestra a continuación:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="ExpEdiciones">
      <xs:sequence>
        <xs:element name="Descripcion" type="xs:string"/>
        <xs:element name="ID" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="EEdicToUpdate">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
        <xs:element name="Update" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="ExpEdiciones_input">
  <wsdl:part name="parameter" type="tns:ExpEdiciones"/>
</wsdl:message>
<wsdl:message name="NewMessageResponse">
  <wsdl:part name="parameter" type="xs:string"/>
</wsdl:message>
<wsdl:message name="UpdateMessageRequest">
  <wsdl:part name="parameter" type="tns:EEdicToUpdate"/>
</wsdl:message>
<wsdl:message name="UpdateMessageResponse">
  <wsdl:part name="parameter" type="tns:ExpEdiciones"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="ExpedienteEdiciones">
  <wsdl:operation name="GenerarExpEdiciones">
    <wsdl:input message="tns:ExpEdiciones_input"/>
  </wsdl:operation>
  <wsdl:operation name="ActualizarEEdiciones">
    <wsdl:input message="tns:UpdateMessageRequest"/>
    <wsdl:output message="tns:UpdateMessageResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="Mi_Binding" type="tns:ExpedienteEdiciones">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="GenerarExpEdiciones">
        <soap:operation soapAction="urn:#NewOperation"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
    </wsdl:operation>
    <wsdl:operation name="ActualizarEEdiciones">
        <soap:operation soapAction="urn:#NewOperation"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
    <wsdl:service name="Exp_Ediciones">
        <wsdl:port name="MiPuerto" binding="tns:Mi_Binding">
            <soap:address location="http://localhost/**"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Después de mostrar las definiciones que componen la descripción del servicio, se puede ver la Figura 3 donde queda conformado de forma general el servicio.

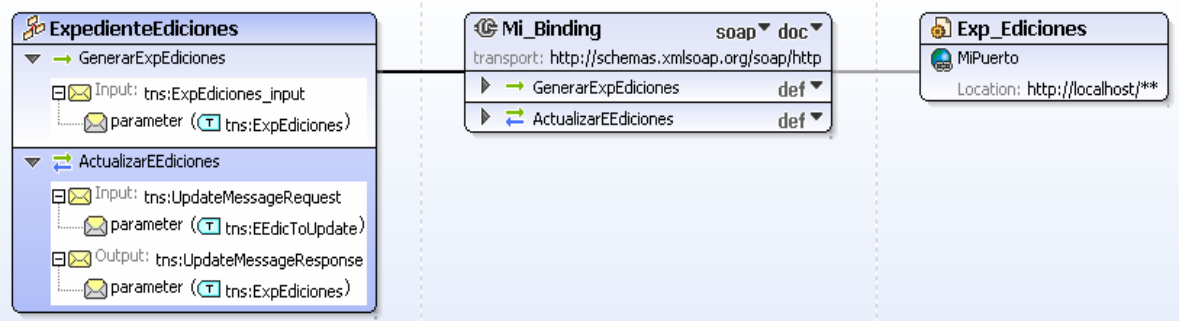

Figura 3. WSDL del servicio Expediente de Ediciones.

Servicio Lista de Matricula:

El servicio Lista de Matrícula del sub-procedimiento de inicio oficial de ediciones de Programa de Maestrías y Especialidades es el encargado de manipular este documento en el sistema. En este caso se decide agregar la operación “Actualizar Lista de Matrícula” que será la encargada de llamar al servicio de aplicación Actualización que mediante su operación “En caso de baja se elimina al estudiante del sistema” actualizará dicho documento. A continuación se muestra como es que se realiza el diseño de este servicio y como queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de datos:

La operación “Se chequean las listas de matrícula para cada profesor” es la encargada de llamar al servicio Buscar para que este busque las listas de matrículas dado el “ID” de un profesor. Para esta operación se crean dos tipos de datos complejos, LMatriculaRequest que contiene el atributo “ID_profesor”, y LMatriculaResponse que contiene “ID” y “Inf_LMatricula” (Figura 4).

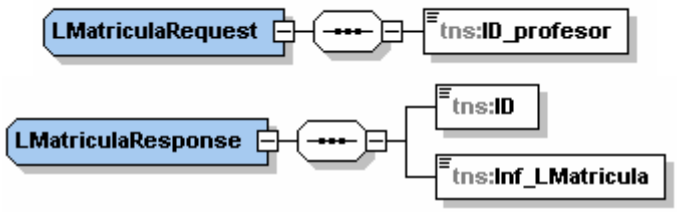

Figura 4. Tipos de datos complejos LMatriculaRequest y LMatriculaResponse.

En el caso de la operación “Actualizar Lista de Matrícula” se crea el tipo de dato complejo Estudiante el cual contiene el atributo “ID” con el fin de identificar el estudiante al que se le quiere dar baja (Figura 5). También se utiliza el ya mencionado LMatriculaResponse.

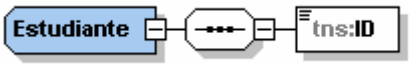

Figura 5. Tipo de dato complejo Estudiante.

El código en el WSDL se muestra a continuación:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="LMatriculaRequest">
      <xs:sequence>
        <xs:element name="ID_profesor" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="LMatriculaResponse">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
        <xs:element name="Inf_LMatricula" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Estudiante">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="ListaMatriculaRequest">
  <wsdl:part name="parameter" type="tns:LMatriculaRequest"/>
</wsdl:message>
<wsdl:message name="ListaMatriculaResponse">
  <wsdl:part name="parameter" type="tns:LMatriculaResponse"/>
</wsdl:message>
<wsdl:message name="UpdateMessageRequest">
  <wsdl:part name="parameter" type="tns:Estudiante"/>
</wsdl:message>
<wsdl:message name="UpdateMessageResponse">
  <wsdl:part name="parameter" type="tns:LMatriculaResponse"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="ListaMatricula">
  <wsdl:operation name="Get_ListaMatricula">
    <wsdl:input message="tns:ListaMatriculaRequest"/>
    <wsdl:output message="tns:ListaMatriculaResponse"/>
  </wsdl:operation>
  <wsdl:operation name="UpdateList">
    <wsdl:input message="tns:UpdateMessageRequest"/>
    <wsdl:output message="tns:UpdateMessageResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="Mi_Binding" type="tns:ListaMatricula">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Get_ListaMatricula">
        <soap:operation soapAction="urn:#NewOperation"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="UpdateList">
        <soap:operation soapAction="urn:#NewOperation"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
<wsdl:service name="L_Matricula">
    <wsdl:port name="MiPuerto" binding="tns:Mi_Binding">
        <soap:address location="http://localhost/**"/>
    </wsdl:port>
</wsdl:service>
```

Después de mostrar las definiciones que componen la descripción del servicio, se puede ver la Figura 6 donde queda conformado de forma general el servicio.

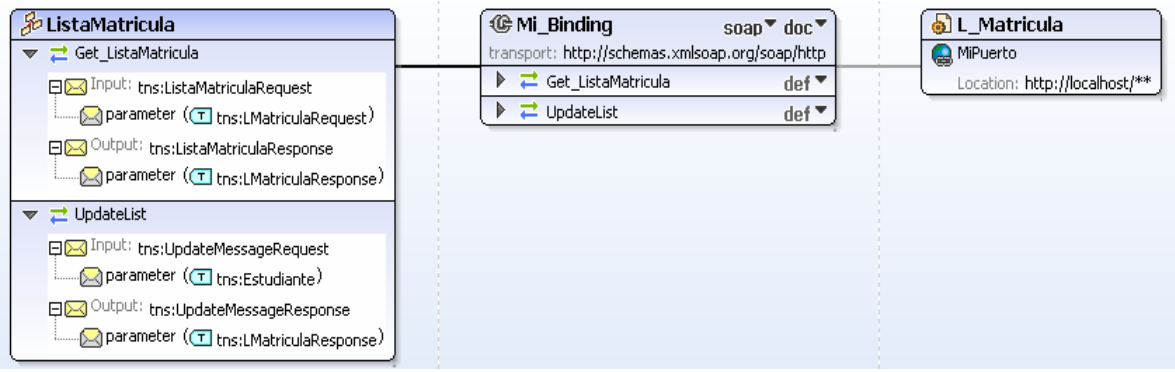

Figura 6. WSDL del servicio Lista de Matrícula.

Servicio Sábana de Notas:

El servicio Sábana de Notas del procedimiento de ejecución y control de las áreas de Maestrías y Especialidades es el encargado de generar este documento en el sistema y consta de una sola operación “Se genera Sábana de Notas”. A continuación se muestra como es que se realiza el diseño de este servicio y como queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de datos:

Para la operación “Se genera Sábana de Notas” se decide crear el tipo de dato complejo Sabana_Notas que contiene el atributo “Descripción” como se muestra en la Figura 7.

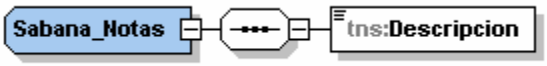

Figura 7. Tipo de dato complejo Sabana_Notas.

A continuación se muestra el código en el WSDL:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="Sabana_Notas">
      <xs:sequence>
        <xs:element name="Descripcion" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="SabanaNotas_Input">
  <wsdl:part name="parameter" type="tns:Sabana_Notas"/>
</wsdl:message>
<wsdl:message name="NewMessageResponse">
  <wsdl:part name="parameter" type="xs:string"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="SabanaNotas">
  <wsdl:operation name="GenerarSabanaNotas">
    <wsdl:input message="tns:SabanaNotas_Input"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="Mi_Binding" type="tns:SabanaNotas">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="GenerarSabanaNotas">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
  </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
<wsdl:service name="S_Notas">
  <wsdl:port name="NewPort" binding="tns:Mi_Binding">
    <soap:address location="http://localhost/**"/>
  </wsdl:port>
</wsdl:service>
```

Después de mostrar las definiciones que componen la descripción del servicio, se puede ver la Figura 8 donde queda conformado de forma general el servicio.

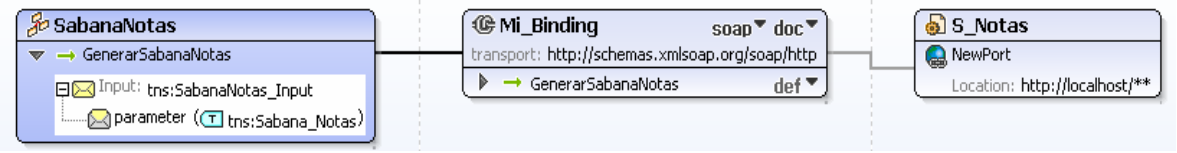

Figura 8. WSDL del servicio Sábana de Notas.

Servicio Sábana de Créditos:

El servicio Sábana de Créditos del procedimiento de ejecución y control de las áreas de Maestrías y Especialidades es el encargado de generar este documento en el sistema y consta de una sola operación “Se genera Sábana de Créditos”. A continuación se muestra

como es que se realiza el diseño de este servicio y como queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de datos:

Para la operación “Se genera Sábana de Notas” se decide crear el tipo de dato complejo Sabana_Notas que contiene el atributo “Descripción” como se muestra en la Figura 9.

Figura 9. Tipo de dato complejo Sabana_Creditos.

A continuación se muestra el código en el WSDL:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="Sabana_Creditos">
      <xs:sequence>
        <xs:element name="Descripcion" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="SabanaCreditos_input">
  <wsdl:part name="parameter" type="tns:Sabana_Creditos"/>
</wsdl:message>
<wsdl:message name="NewMessageResponse">
  <wsdl:part name="parameter" type="xs:string"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="SabanaCreditos">
  <wsdl:operation name="GenerarSabanaCreditos">
    <wsdl:input message="tns:SabanaCreditos_input"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="NewBinding" type="tns:SabanaCreditos">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="GenerarSabanaCreditos">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
  </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
<wsdl:service name="S_Creditos">
  <wsdl:port name="MiPuerto" binding="tns:NewBinding">
    <soap:address location="http://localhost/**"/>
  </wsdl:port>
</wsdl:service>
```

En la Figura 10 donde queda conformado de forma general el servicio.

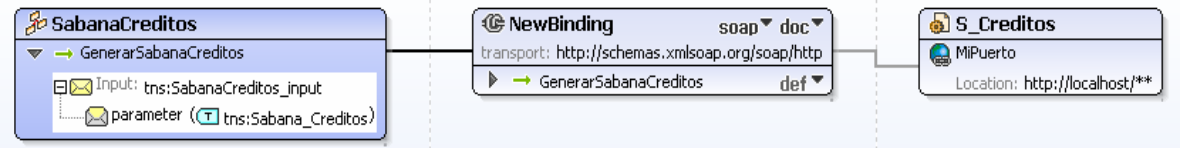

Figura 10. WSDL del servicio Sábana de Notas.

Servicio Acta de Exámenes:

El servicio Acta de Exámenes del sub-procedimiento de inicio oficial de ediciones de Programa de Maestrías y Especialidades es el encargado de generar este documento en el sistema y consta de una sola operación “Se genera Acta de Exámenes”. A continuación se muestra como es que se realiza el diseño de este servicio y como queda conformada la definición abstracta y la definición concreta del mismo.

Definición Abstracta:

Tipos de datos:

Para la operación “Se genera Acta de Exámenes” se decide crear el tipo de dato complejo ActaExam que contiene los atributos “ID” y “Acta_Info” como se muestra en la Figura 11.

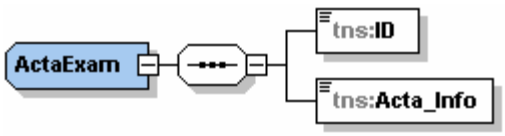

Figura 11. Tipo de dato complejo Sabana_Creditos.

A continuación se muestra el código en el WSDL:

```
<wsdl:types>
  <xs:schema targetNamespace="http://new.webservice.namespace" elementFormDefault="qualified">
    <xs:complexType name="ActaExam">
      <xs:sequence>
        <xs:element name="ID" type="xs:string"/>
        <xs:element name="Acta_Info" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="ActExamInput">
  <wsdl:part name="parameter" type="tns:ActaExam"/>
</wsdl:message>
<wsdl:message name="NewMessageResponse">
  <wsdl:part name="parameter" type="xs:string"/>
</wsdl:message>
```

PortType o Interfaz del servicio:

```
<wsdl:portType name="ActaExamenes">
  <wsdl:operation name="GenerarActaExamenes">
    <wsdl:input message="tns:ActExamInput"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición concreta:

Binding:

```
<wsdl:binding name="Mi_Binding" type="tns:ActaExamenes">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="GenerarActaExamenes">
    <soap:operation soapAction="urn:#NewOperation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
  </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
<wsdl:service name="Act_Exam">
  <wsdl:port name="MiPuerto" binding="tns:Mi_Binding">
    <soap:address location="http://localhost/**"/>
  </wsdl:port>
</wsdl:service>
```

La Figura 12 muestra como queda conformado de forma general el servicio.

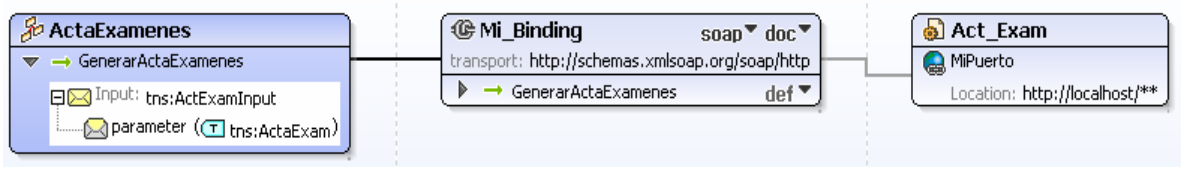

Figura 12 WSDL del servicio Sábana de Notas.