Universidad de Los Andes
Facultad de Ingeniería
Postgrado en Computación
Mérida - Venezuela

MÉTODO PARA EL DESARROLLO DE SERVICIOS WEB

Autor: Ing. Zulma Díaz
Tutor: Dr. Jonás Montilva

Mérida, Septiembre 2006

DEDICATORIA

A Dios Todopoderoso ....mi roca fuerte en todo momento.

A ti hija Anyela, mi Motivo, mi Tesoro... Gracias chiquita por regalarme tu tiempo... Te amo y que Dios te Bendiga por Siempre!!!

A mi Madre, gracias a su apoyo incondicional y por ser manantial de sabiduría espiritual … Te quiero!!!

A mi Padre, por confiar en mi … Te quiero!!!

A ti Hermanita Gladys, por su paciencia, apoyo y amor, por estar a mi lado y valorar mi esfuerzo. Te quiero!!!

A mi Hermana Edelis, por ser un apoyo en todo momento, mil Gracias … Te quiero!!!.

A todos hermanos, porque siempre me han apoyado en todo momento...

A todos mis Sobrinos, para que este logro sea un ejemplo a seguir, ustedes son los responsables de su destino...Sigan adelante!

AGRADECIMIENTOS

A mi Tutor Dr. Jonás Montilva, por compartir sus conocimientos en el transcurso de la investigación, por sus enseñanzas, dedicación, ayuda y confianza, en todo momento, orgullosa me siento de haber trabajado bajo su tutoría. Gracias...

A la UNEG, por darme esta oportunidad de estudios. Gracias.

A la ULA, por darme doble oportunidad de finalizar el postgrado. Gracias.

A mis profesores de postgrado ULA: Jacinto Dávila, Jonas Montilva, José Aguilar, Judith Barrios Wladimir Rodríguez, compañeros y todas aquellas personas presentes en el camino, que de alguna manera aportaron al logro de ésta meta.

A los profesores Juan Guerrero, Juana Ordaz y Henry Izquierdo, por sus sabias sugerencias y aportes.

Al profesor Mauricio Paletta, por su disposición de brindarme su asesoría. Gracias.

A la Sra. Mirian, por la dedicación de su tiempo en las correcciones del trabajo.

A Tupamaro, por brindarme su apoyo en todo momento.

A todo el personal que labora en la Dirección de Informática de la UNEG, muchas gracias por su apoyo en todo momento que los necesité.

RESUMEN

Los Servicios Web (SW) representan una oportunidad de alcanzar la interoperabilidad de los sistemas actuales. Ellos permiten que se continúe utilizando los sistemas legados de las organizaciones y que estos, a su vez, se comuniquen con otras organizaciones, utilizando Internet como canal de comunicación. Desde el punto de vista de la Ingeniería del Software, es importante dotar de mecanismos adecuados, para que la realización de *Servicios Web* satisfaga las necesidades, tanto de los usuarios como de clientes que contratan el servicio o empresas que desarrollan. Sin embargo, en la actualidad no existe un método universalmente aceptado que guíe el proceso de desarrollo e integración de SW. Actualmente, para el desarrollo de Servicios Web, se están utilizando metodologías que son: orientadas a aplicaciones Web, propietarias, ágiles, informales, entre otras, que no especifican lo relacionado con la Arquitectura Orientada a Servicios (SOA). El trabajo de investigación que se desarrolló estuvo dirigido a crear un método que guíe el desarrollo de Servicios Web, centrado en tres elementos metodológicos: un modelo del producto, un modelo de procesos y un modelo de grupo. La metodología de investigación que se utilizó para diseñar el método de SW se basan en principios, modelos y conceptos tomados de la Ingeniería de Métodos (ME) y la Ingeniería del Software donde se emplearon los tres elementos metodológicos: Modelo de producto, modelo de proceso y modelo de grupo. El Método desarrollado fue denominado DESWeb y abarca desde la fase de especificación hasta el despliegue del SW. Entre las bondades del método DESWeb se destacan las siguientes: (1) está basado en la arquitectura SOA; (2) es un punto de partida para que las pequeñas y medianas organizaciones puedan iniciarse en los SW en un muy poco tiempo y con la oportunidad de ofrecer calidad de servicio; (3) consta de tres elementos metodológicos: Modelo de producto, el modelo de procesos técnicos y el modelo de grupo de desarrollo, que se traduce en la interrelación de personas, productos y procesos; y (4) emplea diferentes formas de aprovisionamiento de los componentes que integran un SW, el cual puede ser desarrollado, adaptado o contratado. El método fue validado mediante el desarrollo completo de un sencillo SW.

Palabras clave: Servicios Web, Arquitecturas basadas en servicios, Desarrollo de Componentes de Software Reutilizable

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

INTRODUCCIÓN

1.1 Introducción

Los servicios basados en tecnología web surgen recientemente como un área de interés para la prestación de servicios informáticos distribuidos en diferentes tipos de negocios. Este concepto constituye un importante paso en el diseño de plataformas tecnológicas abiertas donde se agrupan los proyectos que hacen posible la interconexión transparente entre distintos sistemas, de una o diferentes organizaciones, para poder intercambiar datos o realizar transacciones a través de Internet. Es un concepto perteneciente al campo de la computación distribuida y, como tal, apunta a resolver el problema de integración entre distintas plataformas, sin plantear la unificación de las mismas; por el contrario, plantea crear las interfases que hacen posible la integración, pero manteniendo las particularidades.

Los servicios web están diseñados como plataforma de tecnología abierta que lleva a cabo un servicio concreto y que puede integrarse o comunicarse de manera transparente a otras aplicaciones web para brindar otro servicio diferente o más complejo. De ahí que sean considerados como tecnología principal para los procesos de encapsulado de negocio, proporcionando la base para reducir la integración entre ellos y para crear nuevas relaciones.

Por lo antes expuesto, se evidencia el inicio de nuevos retos en cuanto a la construcción de aplicaciones, composición de servicios independientes y preexistentes, instalación de servicios en plataformas diferentes, accesibilidad y desempeño dadas las necesidades aplicativas y tipos de usuarios diferentes. Igualmente se debe nombrar el método que sirva de guía en el diseño y construcción de dichos servicios. Dada la ausencia de herramientas metodológicas apropiadas y

adaptadas a los servicios web; la creación de un método basado en modelos de la Ingeniería de Métodos (modelo de productos, de procesos y de equipo), que se fundamente en arquitecturas orientadas a servicios, brinda una significativa contribución para todas aquellas organización interesadas en la implementación de una infraestructura de SW.

Este trabajo de investigación tuvo como objetivo principal la creación de un método que guíe a un grupo de profesionales en el desarrollo de servicios web.

Este documento está organizado y elaborado siguiendo las normativas de la Universidad de los Andes - Postgrado en Computación para la entrega de proyectos de Trabajo de Grado y respetando los lineamientos de las Normas de publicación de trabajos de la American Psychological Association (APA) en su "Publication Manual" (Washington, 1994). Está estructurado de la siguiente manera:

El segundo capítulo está relacionado con el marco conceptual donde se describen definiciones, relacionadas con: SW, ingeniería web, ingeniería de software, método, metodología y principios de ingeniería del método.

El tercer capítulo contempla la descripción del Método de Desarrollo de Servicios Web (DESWeb).

El cuarto, quinto y sexto capítulos están relacionados con el modelo de producto, modelo de equipo de trabajo y modelo de procesos, respectivamente.

El último capítulo se refiere a las conclusiones del trabajo.

1.2 Definición del Problema

La aparición de los Servicios Web (SW) proporciona la explotación de otros mercados y servicios antes impensables, tales como la integración de plataformas de hardware y sistemas operativos heterogéneos de la misma organización y de otras

organizaciones externas. Los SW han llevado a un importante crecimiento en el desarrollo del software sobre dicha tecnología. Desde el punto de vista de la Ingeniería del Software, es importante disponer de mecanismos metodológicos adecuados, para que la realización de *Servicios Web* satisfaga las necesidades, tanto de los usuarios como de clientes que contratan el servicio o empresas que desarrollan. Sin embargo, en la actualidad no existe un método universalmente aceptado que guíe el proceso de desarrollo e integración de SW. No obstante, existe diversidad de metodologías orientadas a objetos, aplicaciones web, ágiles, propietarias, informales, entre otras que orientan el desarrollo de SW, pero no especifican lo relacionado con la Arquitectura Orientada a Servicios (SOA).

Ante esta necesidad, fue planteado como objetivo del trabajo de grado, construir un método que guíe el desarrollo de SW, centrado en tres elementos metodológicos: un modelo del producto, un modelo de procesos y un modelo de grupo. Esto para desarrollar infraestructuras y soluciones de integraciones flexibles, modulares y basadas en estándares arquitectónicos que favorezcan la reutilización de software.

1.3 Objetivos de la investigación

1.3.1 Objetivo General

Construir un método que guíe a un grupo de especialistas en el desarrollo de SW y de aplicaciones basadas en la composición de este tipo de tecnología.

1.3.2 Objetivos Específicos

1. Caracterizar la terminología asociada con los servicios: WEB *services, e-services, e-commerce, e-business, database services, entre otras*

2. Estudiar las arquitecturas, componentes y protocolos de los SW (SOA, SOAP, XML, HTTP, UDDI, WSDL) existentes para la construcción de este tipo de tecnología.

3. Analizar los modelos de composición de SW.

4. Elaborar un método de desarrollo de SW basado en tres elementos metodológicos: Modelo de producto, modelo de procesos y modelo de grupo.

5. Validación del método mediante un caso práctico.

1.4 Hipótesis

Se puede construir un método para desarrollar SW, que incluya las definiciones de: modelo de producto, modelo de proceso y modelo de grupo de trabajo, mediante el cual pequeñas y medianas empresas puedan desarrollar SW que le permitan unir diversidad de plataformas e interactuar con otras organizaciones de su interés.

1.5 Justificación de la investigación

El hecho de ser los SW una tecnología relativamente nueva trae consigo una serie de brechas que atender; una de ellas es la ausencia de un método. En la práctica se menosprecia el valor de un método para crear el software. Sin embargo, es de vital importancia contar con un método que guíe a un equipo de desarrollo en los pasos a seguir en la elaboración de este tipo de infraestructura. El objetivo de un método orientado a servicios no sólo es generar una mayor productividad en el desarrollo, también, es crear mejores productos.

La situación actual hay que verla como una oportunidad donde las organizaciones, al querer implementar este tipo de tecnología de SW, sean guiadas

por un método que dé como resultado un producto final de alta calidad; y, de manera generar una ventaja competitiva frente a las demás opciones. Por lo especificado anteriormente, se justifica plenamente la investigación, dada la ausencia de un método que guíe paso a paso el desarrollo de SW, relacionado con SOA.

1.6 Metodología

El diseño de la metodología se basan en principios y conceptos tomados de la Ingeniería de Métodos (ME) y de la Ingeniería de Software (SE). Según Odell (1996), un importante principio de la ME establece que un método bien definido se debe describir en términos de dos elementos: un modelo de producto, el cual será desarrollado y un modelo de proceso que explica cómo desarrollar el producto. Fue adicionado un tercer elemento por Montilva y Barrios (2003) en los elementos metodológicos del método Watch, que se refiere al modelo del equipo de desarrollo, el cual describe los roles de cada uno de los miembros que intervienen durante el desarrollo del producto.

Para describir la metodología de investigación que se utilizó para diseñar el método de SW, se elaboró el cuadro mostrado en la Tabla 1.1 que especifica las actividades y herramientas a utilizar para construir cada modelo del método.

Tabla 1.1: Fases y actividades para la construcción de los modelos del método SW(a).

Fase	**Actividades**	**Productos**
Diseño del Modelo del Producto	1. Estudiar la arquitectura orientada a servicios (SOA). 2. Estudiar los protocolos y servicios de SOA. 3. Analizar como interactúan las capas de este tipo de arquitecturas. 4. Diseño de arquitectura basadas en SOA. 5. Diseñar los componentes de cada capa de la arquitectura: Vista, reglas de negocio y servicios de datos.	1. Modelo conceptual de la arquitectura SOA y sus componentes. 2. Modelo conceptual de un SW. 3. Modelo conceptual de un aplicación basada en la reutilización de SW.

Tabla 1.1: Fases y actividades para la construcción de los modelos del método SW(b).

Fase	**Actividades**	**Productos**
	6. Determinar los patrones relacionados: Interfase de Negocio (Business Facade), Reglas de Negocio (Business Services).Determinar las plataformas existentes y cómo pueden acoplarse. 7. Determinar la plataforma existente en la organización con la finalidad de acoplarla a una plataforma que trabaje con SW.	
Diseño del Modelo del Proceso	1. Procesos Gerenciales: Determinar las actividades que se refieran al proceso de administrar el proyecto, asegurando la calidad de SW, riesgos, y el entrenamiento del equipo que trabajará en el desarrollo del SW. 2. Procesos técnicos: Determinar las actividades que son necesarias para modelar el dominio de SW, especificar requisitos, diseño, código, prueba, y entrega de la plataforma de SW. Igualmente definir los servicios y protocolos utilizados en la arquitectura, especificar e identificar los componentes, interacción de componentes.	1. Modelo de procesos gerenciales para el desarrollo de SW. 2. Modelo de procesos técnicos para el desarrollo de SW.
Diseño del Modelo de Actores	Definir los roles de las personas que trabajarán en el desarrollo de los SW.	Modelo de actores para el desarrollo de SW.

1.7 Descripción de los resultados esperados

Estudio de la tecnología de SW (componentes y protocolos): Determinar los estándares que permiten la creación, registro, ubicación y uso de los SW y las relaciones de las tecnologías, considerando al proveedor de SW desde que coloca a disposición un SW hasta la invocación del mismo por cualquiera consumidor que quiera hacer uso del SW.

Estudio y caracterización de los SW: plasmar de manera sencilla los conceptos resaltantes relacionados con los SW.

Diseño y documentación de un método para el desarrollo de SW: la documentación producto del diseño con miras a facilitar de manera sistemática el desarrollo de SW

Validación experimental a través de la creación de SW (demostración): con la finalidad de probar el método, a través de un caso práctico que sirva de ejemplo a las empresas interesadas en desarrollar este tipo de tecnologías.

CAPÍTULO II

MARCO CONCEPTUAL

En este capítulo se describen los conceptos relacionados con la arquitectura de los SW y sus diferentes modelos y aspectos tecnológicos: XML, SOAP, WSDL y UDDI.

Igualmente, se definen los llamados Proveedores de Servicios de Aplicaciones (ASP), definición de SW, composición de SW, que refieren los términos orquestación y coreografía, utilizados para definir aspectos relacionados con la creación de procesos de negocios que involucran varios SW y la arquitectura de una aplicación genérica de SW.

A la par se presentan definiciones, tales como ingeniería web, ingeniería de software, método, metodología y principios de ingeniería del método.

2.1 Arquitectura de los SW.

Una Arquitectura Orientada a Servicios (SOA - *Service Oriented Architecture)* está compuesta por tres partes: un proveedor, un intermediario y un cliente, lo cual permite la existencia de acoplamiento entre ellos. De acuerdo con W3C (2004), la arquitectura de un SW (WSA – *Web Service Architecture*) es un ejemplo de implementación de la Arquitectura Orientada a Servicios (SOA), que está basada en servicios. SOA es una meta-arquitectura que permite que ciertos servicios de red sean descritos, publicados, descubiertos e invocados en un ambiente de cómputo distribuido de forma dinámica.

Los SW están constituidos por tres componentes (ver Figura 2.1) – dado que se acoge a la SOA; y ésta esta compuesta de tres elementos - : ***Proveedor del servicio***, es el que suministra el servicio que otras organizaciones puedan utilizarlos***; Registro de servicios***, es el repositorio de descripciones de servicios donde cualquier ente puede accederlo y el ***Solicitante de servicios***, es el que hace uso del servicio para

dar solución a una situación. La Figura 2.2 muestra un diagrama de clases de los componentes de un SW.

Igualmente, se mencionan las operaciones presentes en los SW: ***publicar,*** labor que realiza el proveedor del servicio al registro con el objetivo de colocar disponible el servicio al público; ***buscar,*** cuando el cliente hace la exploración en el repositorio de servicios con la finalidad de encontrar un servicio que cumpla con sus necesidades. Por último tenemos el término ***encontrar***, que permite acceder a un SW con el fin de esperar un resultado.

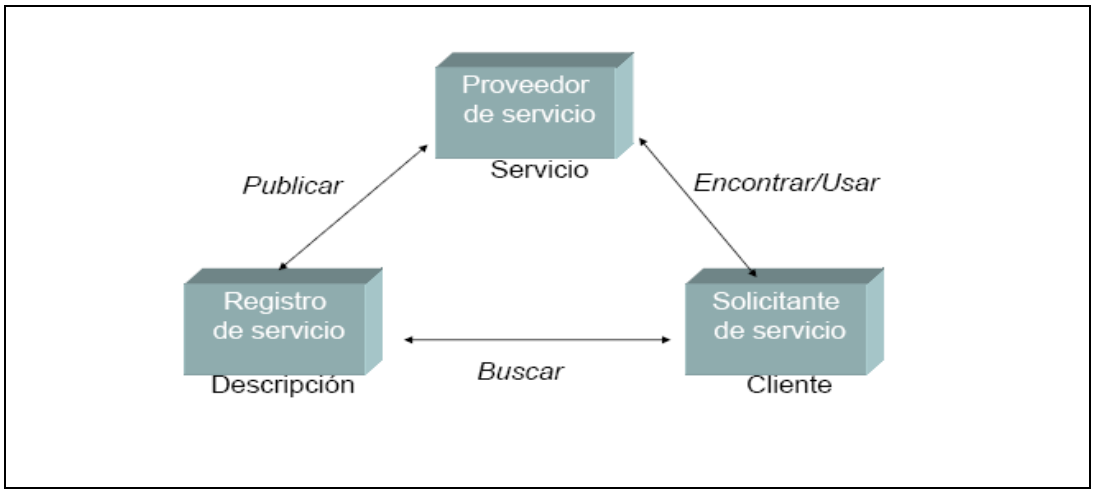

Figura 2.1: Componentes de SW y sus relaciones.
Fuente: http://www.fisica.uson.mx/carlos/WebServices/WSRevolution.htm

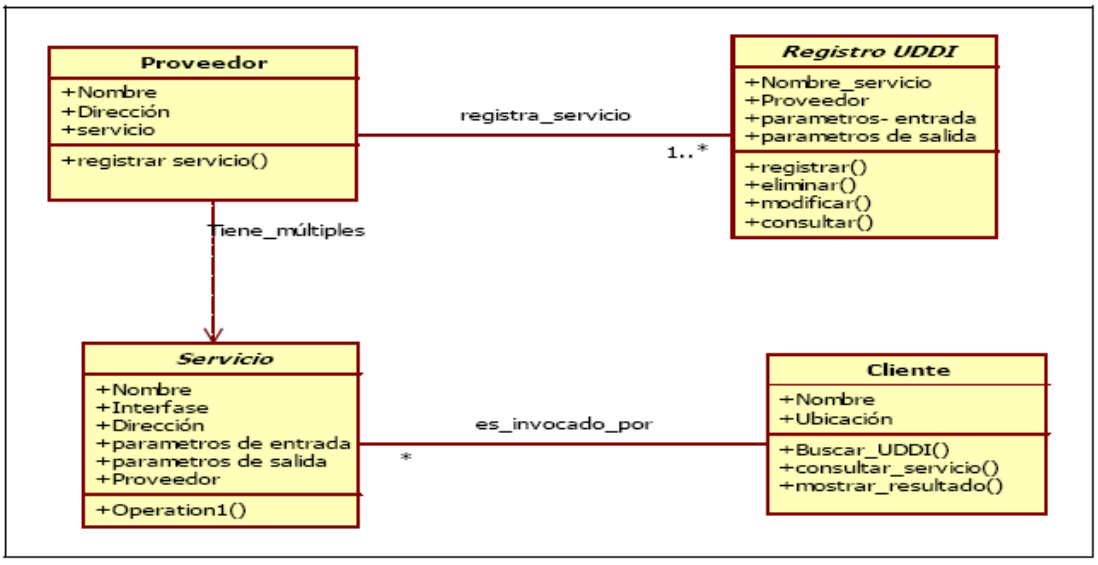

Figura 2.2: Diagrama de Clases de los componentes de un SW.

Según W3C (2004) la arquitectura de los SW se compone de cuatro modelos: Modelo Orientado a Mensaje (MOM), Modelo Orientado a Servicio (SOM), Modelo Orientado a Recursos (ROM) y Modelo de Políticas (PM).

Hay dependencias entre los cuatro modelos, como se muestra en la Figura 2.3: SOM depende de MOM; MOM depende de PM; ROM depende de SOM. Cada modelo se etiqueta con su concepto dominante.

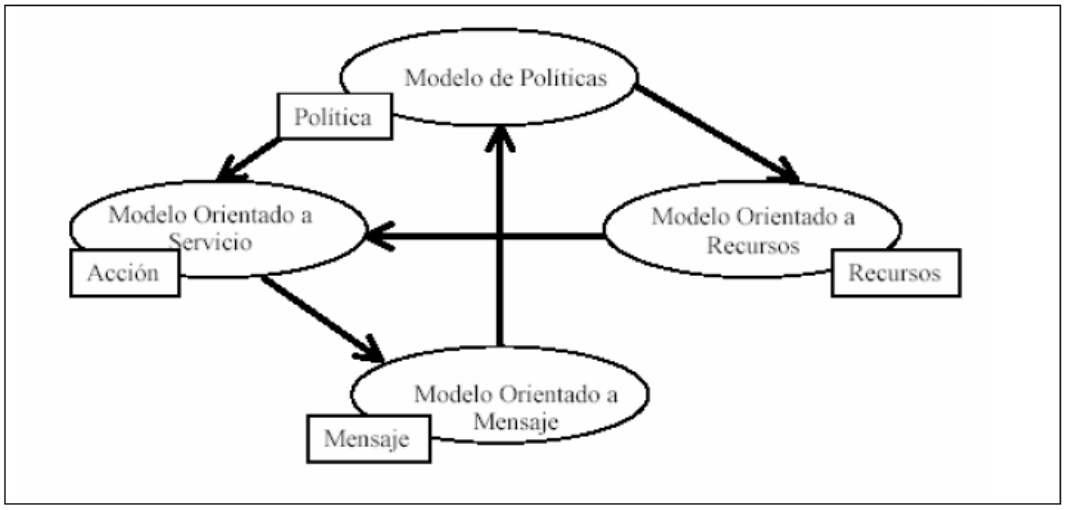

Figura 2.3: Modelos de la arquitectura de SW .
Fuente: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

2.1.1 Modelo Orientado a Mensaje (MOM).- Está orientado al mensaje, estructura, transporte, entre otras. Su función principal gira alrededor de los conceptos remitentes, receptor de mensajes, estructura del mensaje en términos de encabezados y cuerpo de mensaje, así como los mecanismos utilizados para ejecutar su entrega. Existen detalles adicionales a considerar, como las funciones de políticas y cómo es manejado el modelo a nivel de mensaje. El MOM se centra en aspectos de la arquitectura relacionada con mensaje y procesamientos de los mismos.

2.1.2 Modelo Orientado a Servicios (SOM).- Está enfocado en los conceptos de servicio y acción dentro de la arquitectura de SW. El propósito principal del SOM son las relaciones entre entidad y los servicios que proporcionan y solicita. El SOM es contrario a MOM, dado que se enfoca en la acción más que al mensaje.

2.1.3 Modelo orientado a recursos (ROM).- Se centra en los aspectos de la arquitectura que relaciona con los recursos. Es un concepto principal sobre el cual está fundamentada la Web y los SW; considerando un SW como un tipo de recurso.

ROM se enfoca en describir los aspectos fundamentales de los recursos, independientemente de las funciones que estos tengan en el contexto de los SW. Describiendo sus propiedades, políticas, entre otras. En tal sentido los SW son recursos y estas características son heredadas por estos.

2.1.4 Modelo de Políticas (PM).- El PM está orientado a aquellos aspectos de la arquitectura que se relacionan con la política de seguridad y calidad del servicio. La seguridad está fundamentada sobre restricciones, acción y recursos a que tienen acceso. Igualmente, la calidad del servicio está sustentada sobre estas restricciones. En el PM, estas restricciones se modelan alrededor del concepto de las políticas y las relaciones con otros elementos de la arquitectura.

2.2 Proveedor de Servidores de Aplicaciones (*Application Server Providers*, ASPs)

Los ASP son empresas que se dedican a desarrollar sistemas integrados utilizando tecnologías web a lo que se le denominan "Planificación de Recursos Empresariales (*Enterprise Resource Planning*, *ERP*), que consiste en utilizar información de áreas tan diversas de las organizaciones como administración de recursos humanos, finanzas, compras, fabricación, logística y distribución, ventas, mercadeo y relaciones con clientes, proporcionando una forma fácil e integral de acceder, ver, y utilizar la información de todas las áreas funcionales de la organización, utilizando una sola aplicación, para alquilar a terceros como un servicio mediante un contrato, que es accedido desde las instalaciones de la empresa proveedora. La organización que utiliza el servicio es dueña sólo de la información que suministra la plataforma.

Igualmente, Wikipedia (2006) expone que "Un ***Proveedor de Servicios de Aplicaciones*** (PSA; en inglés *application service provider*, ASP) es una empresa que proporciona servicios de software a múltiples entidades desde un centro de cómputo a través de una red. Entre los factores que habilitan a un PSA se destacan la amplia difusión del uso de Internet, la capacidad de acelerar el despliegue y implementación de aplicaciones y la posibilidad de transferir servicios y operaciones a terceros. La barrera principal para un PSA radica en convencer a sus clientes de que su información en manos de un tercero permanece segura".

Estas aplicaciones son genéricas pero desarrolladas considerando la capacidad de particularidad y modularidad. Tarsis (2006) comenta que el modelo PSA trae consigo una serie de beneficios a las pequeñas y medianas empresas:

1. No se paga por el desarrollo de la aplicación, sino por el uso y particularización de una ya creada. Lo que significa que el costo será menor ya que no se paga por un desarrollo completo ni actualización de las aplicaciones.

2. Se puede empezar a trabajar mucho antes con ellas, al tener que ser solamente particularizadas en su aspecto para ajustarse a la imagen de su organización.

3. La empresa proveedora se ocupa del mantenimiento de las aplicaciones.

4. Las aplicaciones están sometidas a una permanente mejora, que se incorpora al servicio que se contrata sin costo adicional.

5. Las aplicaciones son modulares, por lo que puede aprovechar el alojamiento que ha contratado en su primer servicio para alojar un segundo o tercero a un precio más económico.

Igualmente PC- NEWS (2002) considera que de PSA expone toda una gama de aplicaciones usadas en una organización, desde el uso de una aplicación remotamente, un sistema ERP (*Enterprise resource planning*), CRM (*customer*

relationship management), distribuciones masivas de información, hasta servicios de hospedaje, telefonía sobre IP, soporte remoto, entre otras.

De acuerdo con Bussler (2002) los PSA son un nuevo tipo de servicio proveedor que hace su aparición desde aplicaciones pequeñas como E-mail hasta aplicaciones para grandes empresas que pueden ser invocadas por clientes desde la web. El proveedor de las aplicaciones se encarga de instalar y manejar las aplicaciones en su propio centro de datos. En su lugar, el cliente paga un honorario para tener acceso al uso sobre el Internet. Un PSA administra las aplicaciones así como mejoras a las mismas. Un componente importante del servicio de un ASP es el respaldo de los datos y aplicaciones y apoyo en la recuperación de datos cuando sea necesario.

Considerando las definiciones anteriores relacionadas con PSA, se puede concluir que los PSA son empresas que básicamente alquilan este tipo de aplicaciones, de manera que las empresas deleguen a organizaciones externas la gestión de aplicaciones de su organización para eliminar los costos que supone lo relacionado con: tecnologías de la información, compras de licencias, entrenamiento de personal, entre otras.

2.3 Definición de Servicio Web (Web Service)

Por servicio web entendemos el uso de tecnología Internet (web, mail, ftp, entre otras.) para interconectar procesos, generalmente transaccionales, entre empresas y sus clientes o proveedores. Estos sistemas son del tipo extranet /intranet, ya que no están accesibles al público general (Ramírez, 2005). Igualmente, son considerados como tecnología emergente, impulsada por el deseo de exponer de forma segura la lógica de negocios en Internet. Haciendo uso de los SW las organizaciones pueden encapsular sus procesos de negocios, publicarlos como servicios, suscribirse a otros servicios e intercambiar información con otras organizaciones.

Nandigam (2005) considera a un SW como una unidad interoperable de la lógica de una aplicación que es independiente de lenguajes de programación, sistemas operativos, protocolos de comunicación de red, y dependencias y ediciones de la representación de datos. Es una infraestructura para desarrollar y desplegar aplicaciones distribuidas.

Antes de dar una definición propia sobre el significado de SW, se puede dar una definición de las palabras que la conforman; *Servicio* es un medio que expone su funcionalidad utilizando una interfaz programática, lo que significa que fue elaborado para ser utilizado por una aplicación (*software*), más bien que por los seres humanos, la forma de accederlo y de las respuestas posibles están plasmadas mediante un documento. *Web* es esencialmente un universo de información accesible como texto, gráficos y otros objetos multimedia, utilizando Internet.

Tomando las definiciones anteriores, se puede inferir que los *SW,* son componentes de software distribuidos, públicos y reutilizables, basados en estándares de Internet, que comprenden proyectos mediante los cuales es posible la interconexión entre distintos sistemas, de una o diferentes organizaciones, con la finalidad de intercambiar datos o comercializar funciones de negocio, utilizando la web; y que ofrece una funcionalidad concreta, independiente tanto del lenguaje de programación donde fue implementado como de la plataforma de ejecución.

Los SW vienen a dar solución al problema de integración entre variedad de plataformas, sin considerar la unificación de las mismas; más bien implantar interfases que hagan posible la integración, pero conservando las peculiaridades.

Los SW son aplicaciones independientes de la plataforma que pueden ser fácilmente publicadas, localizadas e invocadas, a través de protocolos web estándar: XML(*eXtensible Markup Language*), SOAP(*Simple Object Access Protocol*), UDDI(*Universal Description, Discovery and Integration*) y WSDL(*Web Services Description Language*).

2.4 Aspectos tecnológicos de los SW

Para realizar las operaciones que involucren los SW, es necesario que existan estándares en los distintos niveles que comprenden. Los SW se registran y anuncian utilizando los siguientes servicios y protocolos.

2.4.1 XML (Extensible Markup Language)

XML es el estándar de Extensible Markup Language. XML no es más que un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes. XML es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructuradas (Topxml, 2005). Está basado en el anterior estándar SGML (*Standard Generalized Markup Language*, ISO 8879), que data de 1986, pero que empezó a gestarse desde principios de los años 70, y a su vez basado en el GML creado por IBM en 1969. Esto significa que aunque XML pueda parecer moderno, sus conceptos están más que asentados y aceptados de forma amplia. Está igualmente asociado con la recomendación del W3C DOM (*Document Object Model),* aprobado también en 1998.

XML se derivó de SGML como subconjunto simplificado, eliminando las partes más engorrosas y menos útiles. XML es un metalenguaje: es un lenguaje para definir lenguajes. Los elementos que lo componen pueden dar información sobre lo que contienen, no necesariamente sobre su estructura física o presentación, como ocurre en HTML.

2.4.2 SOAP (Simple Object Access Protocol)

SOAP es un protocolo utilizado para el intercambio de la información en un ambiente descentralizado - distribuido. Es un protocolo basado en XML que consiste en: definición de un marco para describir cuál es el mensaje y cómo procesarlo, un sistema de reglas de codificación para expresar casos de tipo de datos de definición de aplicaciones, y una convención para representar llamadas y respuestas a procedimientos remotos (W3C, 2000).

SOAP es un protocolo de envío de mensajes entre clientes y servidores Web. SOAP contiene también varias secciones opcionales que describen las llamadas a métodos (RPC) y proporcionan detalles sobre el envío de mensajes SOAP a través de HTTP. El mensaje SOAP consta de cuatro partes: ***envoltura (envelope)*** que es la raíz del formato SOAP, ***encabezado*** (*header*) es el elemento opcional, cuya finalidad es ampliar las funciones del protocolo, ***cuerpo***(*body*) contiene los datos del mensaje, es obligatorio, y ***error***(*default*) contiene información de ocurrir un error. La Figura 2.4 muestra la estructura de un mensaje SOAP y la Figura 2.5 visualiza la invocación y respuesta de un SW.

Figura 2.4: Estructura de un Mensaje (SOAP).

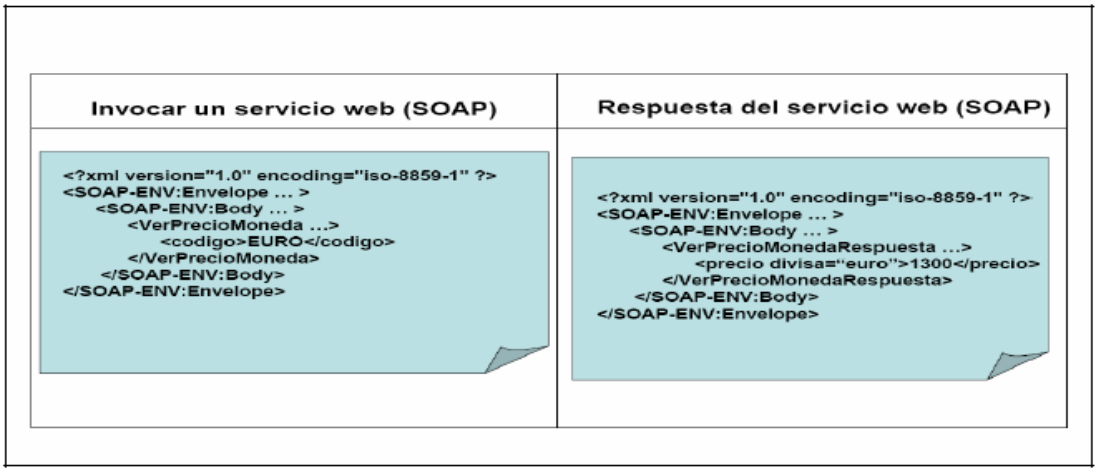

Figura 2.5: Invocación y respuesta de un SW.

Erl (2004), señala que la especificación del SOAP establece un formato de mensaje estándar que consiste de un documento XML capaz de recibir documentos de la clase RCP y documento-céntricos. Esto facilita intercambio de modelos de datos

síncrono (petición y respuesta) así como asíncrono (proceso-manejado). Con WSDL se establece un formato estándar de descripción de punto final para las aplicaciones, el formato de mensaje de documento-céntrico es mucho más común.

2.4.3 WSDL (*Web Service Description Language*)

WSDL es lenguaje basado en XML que describe los servicios de red como un conjunto de puntos finales que procesan mensajes contenedores de información orientada, tanto a documentos como a procedimientos. Las operaciones y los mensajes se describen de forma abstracta y después se enlazan a un protocolo de red y a un formato de mensaje concreto para definir un punto final de red. Los puntos finales concretos relacionados se combinan en puntos finales abstractos (servicios). WSDL es extensible, lo que permite la descripción de puntos finales de red y sus mensajes, independientemente de los formatos de los mensajes o protocolos de red utilizados para comunicarse (Microsoft, 2003).

Para Short (2002), WSDL es un dialecto basado en XML sobre el esquema que describe un SW. Un documento WSDL proporciona la información necesaria al cliente para interaccionar con el SW.

WSDL es utilizado para describir totalmente el SW, detallar su ubicación y describir las operaciones y métodos que él presenta. Para ello especifica cómo crear diferentes asociaciones a partir de un esquema XML: tipos → mensajes →, operaciones → vínculos a protocolos de transporte → localización del servicio.

Cuando el desarrollo de un SW ha finalizado, el siguiente paso es el despliegue y su respectiva publicación mediante un documento WSDL que consiste en la construcción de enlaces en un repositorio UDDI para que los usuarios potenciales puedan hacer uso del mismo. Al momento de su utilización, se hace uso del archivo WSDL para conocer la ubicación del servicio, llamado de funciones, y cómo acceder al SW. Luego se utiliza la información contenida en el archivo WSDL,

para construir una petición SOAP (*Simple Object Access Protocol*) y enviarla hacia el proveedor de servicio.

La estructura de un documento WSDL es: *porType,* donde se especifican las operaciones proporcionadas por el SW, *message:* se detallan los mensajes usados por el SW, *types*: describen los tipos de datos utilizados por el SW, y *binding:* se especifican los protocolos de comunicaciones usadas por el SW.

Un documento WSDL tiene una estructura similar a la siguiente:

```
<definitions>
        <types>
                tipos de datos...
        </types>
        <message>
                definiciones del mensaje...
        </message>
        <portType>
                definiciones de operación ...
        </portType>
        <binding>
                definiciones de protocolo...
        </binding>
</definitions>
```

2.4.4 UDDI (Universal Description, Discovery and Integration)

UDDI es el repositorio de los SW. Su función principal es simplificar la descripción y la publicación de servicios, de parte de organizaciones que desean que otras entidades utilicen su servicio, y su descubrimiento e integración por parte de aquellas que requiere de su uso. Los servicios se ofrecen a través de la Web. Es usado para facilitar la comunicación a través de protocolos, tales como SOAP. SOAP es un estándar de operación entre plataformas que proporciona a las organizaciones y aplicaciones una manera de encontrar y usar de forma fácil, eficaz y dinámica los SW a través de Internet. También permite la administración de registros de operación para ser usados en cualquier contexto operacional. El estándar UDDI provee un

mecanismo para que las empresas describan sus negocios y los tipos de servicios que ellas proporcionan, además registrar y publicar en un registro UDDI. Los servicios ya publicados pueden ser buscados por otras empresas utilizando el protocolo SOAP. Una vez descubiertos los negocios con quien se pueden asociar, pueden utilizan este mecanismo para integrar sus servicios en conjunción con sus socios y proveen los servicios a sus clientes (Lizárraga, 2002).

Un registro UDDI es, en sí mismo, un SW. Expone una API basada en SOAP para tener acceso y manipular entradas del directorio. En lugar de mantener datos a través de un registrador, un desarrollador puede programar directamente con la API (Short, 2002).

Los documentos XML guardados en el sistema UDDI son alojados por compañías que aceptan mantener un nodo y siguen la especificación dada por el consorcio UDDI.org. Actualmente, varias son las organizaciones como Microsoft e IBM que mantienen nodos públicos.

Los datos XML-UDDI manejados por los repositorios se dividen en tres categorías:

1. Páginas Blancas: Con información general sobre una empresa concreta (nombre, descripción, información de contacto, dirección y teléfono).

2. Páginas Amarillas: Con datos sobre las compañías y los servicios que ofrecen.

3. Páginas Verdes: Con información técnica sobre un SW. Generalmente esto incluye un apuntador a la especificación externa y una dirección donde invocar el servicio. UDDI puede ser utilizado para describir cualquier servicio, desde una página Web o una dirección de correo electrónico hasta servicios SOAP, CORBA, entre otras.

2.5 Composición de Servicios

Con la llegada de la filosofía de los SW, los términos “Composición de SW” y “Flujo de SW” son utilizados para describir la composición de SW en un flujo de proceso. Según Cubillos (2004), componer servicios significa establecer mecanismos que permitan a dos o más de ellos cooperar entre sí para resolver requisitos que van más allá del alcance de sus capacidades individuales. De ahí que la composición representa la implementación de SW complejos que internamente invocan a otros SW.

Para la creación de los procesos compuestos, donde intervengan varios SW, es necesario utilizar especificaciones estándares (Cubillos,2004), tales como: modelos de interacción más complejos: asíncronos, reactivos basados en eventos, entre otros; la posibilidad de establecer condiciones de sincronización relacionadas con los SW que intervienen; y consideración de posibilidad de fallas al momento de su ejecución, si en la inexistencia de algunos de los servicios involucrados no ésta disponible; de ocurrir una falla, y ya se haya procesado parte de la transacción, se pueda revertir el proceso sin ninguna complicación.

Los conceptos de Orquestación y Coreografía son utilizados para representar aspectos relacionados con la creación de procesos de negocios que involucren varios SW.

2.5.1 Orquestación

La orquestación es la integración de varios SW (internos o externos) con la finalidad de originar nuevos procesos o servicios de mayor nivel para dar solución a una situación y donde el control del proceso de negocio está caracterizado por una de las partes que intervienen en el proceso.

BPEL - (*Business Process Execution Language*) es un lenguaje basado en XML, diseñado para la orquestación de SW. De acuerdo con lo ya definido,

"orquestación" es el control centralizado del acceso a varios SW, incluyendo la lógica de negocio.

Según Weerawarana (2005), BPEL primero fue llamado como BPEL4SW pero pronto se cambió a WS-BPEL, sin embargo es más conocido como BPEL. BPEL proporciona una notación formal de los procesos del negocio y protocolos de interacción de negocios. BPEL extiende el modelo de SW y habilita a soportar transacciones comerciales y define un modelo de integración interoperable que facilita la expansión del proceso de integración, ya sea, a nivel de intra-corporación (dentro de la misma empresa), como, también, en la interacción con el exterior (empresa a empresa).

BPEL es el estándar de proceso de negocio introducido por Bea Systems, IBM y Microsoft; y el estándar propuesto con el resto de organizaciones (como WebMethods) es el *Business Process Modeling Languaje* (BPML)

BPEL es un lenguaje mediante el cual las empresas pueden alcanzar un gran dinamismo con su arquitectura tecnológica y pueden realizar rápidos cambios o adaptaciones internos o externos, logrando de esta manera facilitar la comunicación entre sus aplicaciones.

BPEL define un modelo y una lingüística para poder representar las fases de un proceso de negocio basado en interacciones entre socios o empresas. Las interacciones entre las entidades se realizan a través de interfaces de SW y la estructura de la relación al nivel de interfase se encapsula en los llamados *convenio de socios*. BPEL especifica cómo las interacciones de servicios con socios son coordinadas para lograr un objetivo común, de la misma manera que el estado y lógica necesarias para esta coordinación (Sánchez, 2004).

Igualmente, BPEL establece aquellos mecanismos necesarios en casos que sucedan una situación problemática antes que el proceso llegue a su fin.

2.5.2 Coreografía

Peltz (2003) define a la coreografía como una secuencia de mensajes que pueden involucrar múltiples participantes y múltiples fuentes, incluyendo clientes, proveedores y socios. La coreografía es típicamente asociada con el intercambios de mensajes públicos que ocurren entre los múltiples servicios web, más bien que a un proceso específico del negocio que es ejecutado por un solo participante o parte.

De acuerdo a esta definición, se puede decir que la ***coreografía*** es la colaboración de los participantes involucrados en el proceso de negocio desde un punto de vista global, un desempeño visible, común y complementario; donde ordenado el mensaje, intercambia un resultado conforme a un objetivo de negocios común.

2.5.3 Diferencia entre orquestación y coreografía

De acuerdo con las definiciones dadas anteriormente relacionada con orquestación y coreografía, se puede apreciar que existen diferencias resaltantes entre ellos, mientras la primera hace mención a cómo interactúa un usuario con el servicio, la segunda se refiere a cómo trabaja el servicio desde el enfoque del proveedor del servicio. Sin embargo existen diferencias definidas, tales como:

1. En la orquestación el proceso es privado, mientras que en la coreografía el proceso es público.

2. La orquestación se realiza mediante pasos dentro de un *workflow* y en la coreografía existen secuencias de mensajes observables.

3. En la orquestación, el proceso es controlado por una de las partes involucradas, mientras que en la coreografía se establece una conversación entre las partes.

4. En la orquestación dice “ así debe ser” y en la coreografía dice “ así debe lucir”.

2.6 Arquitectura de una aplicación genérica de SW

Los SW acogen la arquitectura distribuida multi-capa, como base (Chartier, 2001). Las capas principales de este modelo son: la capa de ***usuario***, la capa de ***presentación***, compuesta de interfases, navegación y validaciones simples; la capa intermedia que es la ***lógica del negocio*** donde residen objetos genéricos, especializados y algoritmos, y; una capa de ***datos*** que proporciona la persistencia para entidades, transacciones, entre otras. La Figura 2.6 muestra las capas que integran la Arquitectura de una aplicación genérica de SW.

2.6.1 Capa de Usuario: Está formada por aplicaciones que ejecuta el usuario. Los tipos de usuarios pueden ser muy variados (aplicaciones de escritorio, navegadores web, dispositivos portátiles, entre otros). De acuerdo al usuario, se diferencian dos clases: ***gruesos,*** representados por aquellas aplicaciones que acceden directamente a la capa de negocio, y finos, caracterizados por navegadores web que acceden a la capa de presentación.

2.6.2 Capa de Presentación: Consiste en servidores WEB que manejan requerimientos HTTP y genera dinámicamente código de presentación para su ejecución y despliegue en el cliente, de forma típica en la forma de páginas HTML. Igualmente, puede producir resultados XML, DHTML, WML, entre otros. Muchos de los requerimientos de personalización son manejados en esta capa. El servidor de presentación puede parametrizar la generación de código de presentación del cliente basándose en el ID del usuario, preferencias, roles, afiliación, entre otros.

2.6.3 Capa de Negocio: Tiene como objetivo encapsular la lógica del negocio bajo un modelo de objetos. En esta capa, se implementan las funcionalidades que corresponden a las reglas del negocio y que se obtienen del análisis de requerimientos y conocimiento del negocio

2.6.4 Capa de Datos: Encargada de asegurar la persistencia de los datos y su recuperación eficaz. Es una partición esencial, básicamente, es el sistema de

manejador de base de datos, archivos planos, también incluye los procedimientos almacenados (*stored procedure*).

Ejemplo de la composición de estas capas son: *Presentación:* HTML y Win32, *Lógica del negocio*: MTS, ASP, MSMQ, ACTIVEX y *Datos*: SqlServer, Ado, XML, OLEDB Informix, Oracle, SysBase.

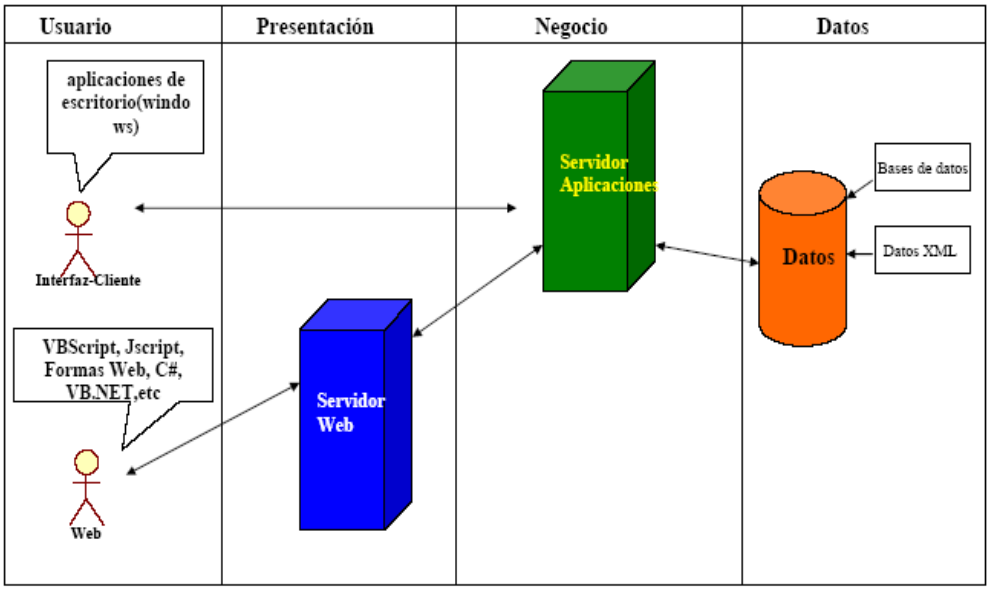

Figura 2.6: Arquitectura de una aplicación genérica de SW.

2.7 Ingeniería Web, ingeniería de software, método, metodología y principios de ingeniería del método

La creciente necesidad de información de las empresas, la necesidad de interconectar diferentes plataformas y el intercambio de información entre organizaciones, tiene su solución en los llamados SW, obviamente esto exige funcionalidad, confidencialidad, *usabilidad* y validez por nombrar particulares de la calidad de software.

El desarrollo de ***"SW "***, al igual que el desarrollo de cualquier software dentro de una empresa, necesita de métodos, técnicas y herramientas para abordar el desarrollo eficaz y eficiente de tales aplicaciones, es necesario tener presente los

conceptos de tales como *"Ingeniería Web",* que engloba cualquier tipo de sistema IT(tecnología de información) o cualquier aplicación donde su interfaz sea a través de la web. Los proyectos de ingeniería web se considera como un proyecto de *"Ingeniería de Software*", cuyo objetivo por décadas ha sido encontrar procesos o metodologías predecibles y repetibles que mejoren la productividad y la calidad **(**Wikipedia, 2005**)**

Igualmente, es propicio también definir lo que es método y metodología. Por método se considera un conjunto ordenado de pasos para alcanzar un resultado, haciendo uso de procedimientos, técnicas y herramientas que permitan a los desarrolladores la producción de software; y por metodología un estudio de una familia de métodos.

De acuerdo con lo afirmado en párrafos anteriores, se evidencia la necesidad de establecer una orientación disciplinada y ordenada para desarrollar un proyecto de software, el cual comprenda de un método acorde que contenga: pasos y procedimientos a seguir, división de etapas o fases, las cuales incluyan actividades y tareas, artefactos a producirse en cada fase, herramientas a utilizar y la gestión y control del proyecto.

Un método, basado en la Ingeniería de Software, es aquel que: (1) establece como emprender de manera ordenada la construcción de software, (2) está centrado en el diseño y modelado; (3) se plantea la situación problemática y su respectiva solución mediante conjuntos de modelos y (4) Describe lo relacionado con la tecnología, proceso y organización. Igualmente, al elaborar un método hay que considerar el principio de la Ingeniería del Método, según Odell (1996), un método comprende los siguientes modelos: producto, proceso y grupo, donde:

Modelo de producto: Describe la arquitectura y los conceptos relacionados con el producto para el cual se está construyendo el método.

Modelo de proceso: Es la definición de los procesos, actividades y tareas que debe realizar el grupo de trabajo con la finalidad dar como resultado el producto establecido.

Modelo de grupo: Es el recurso humano que estará participando en el desarrollo de todas las etapas que den cómo resultado el producto.

CAPÍTULO III

MÉTODO DE DESARROLLO BASADO EN SERVICIOS WEB- SW (DESWEB)

Este capítulo trata sobre los fundamentos del método de desarrollo de SW (DESWeb); es decir, sus bases metodológicas, los modelos y métodos sobre los cuales se diseñó, la descripción de su estructura compuesta del modelo del producto, modelo del proceso y modelo del grupo. Se describen, por consiguiente, los pilares fundamentales que se emplearon en la elaboración del método. Se describen, también, las características particulares del método.

Este método, denominado *DESWeb***,** describe el ciclo de vida de un SW. Método que guía fase a fase el proceso de desarrollo de un SW, desde la especificación hasta el despliegue. Éste método puede ser utilizado en la:

- Creación de nuevos SW.
- Transformación de SW existentes.
- Modificación de aplicaciones existentes en SW.
- Construcción de nuevos SW desde otros SW.

3.1 Fundamentos del Método de Desarrollo basado en SW

La elaboración del método DESWeb se fundamenta en los modelos de: UML Components (Cheesman and Daniels, 2001), método WATCH (Montilva y Barrios, 2003), método WATCH – Component (Hamar, 2003) y el modelo de vista 4 +1 de Philippe Kruchten.

3.1.1 UML Components (Cheesman and Daniels, 2001)

UML Components (Cheesman and Daniels, 2001) se orienta al desarrollo de aplicaciones basadas en arquitecturas de N capas (n-tier). La arquitectura de N capas se encuentra constituida de 4 capas divididas en dos partes: Una parte referida al cliente y la otra parte orientada al servidor donde reside las aplicaciones, ver Figura 3.1.

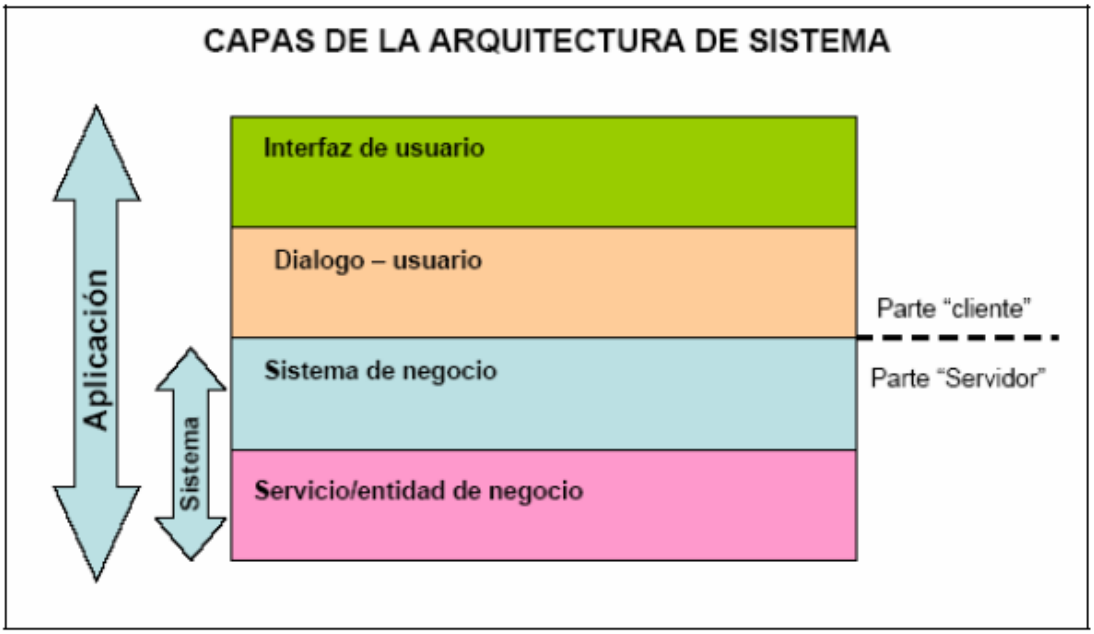

Figura: 3.1 Capas de la arquitectura de sistema.
Fuente: Cheesman and Daniels (2001)

UML Components es un método que está constituido de las fases de requisitos, especificaciones, aprovisionamiento, ensamblaje, pruebas y despliegue.

Siguiendo la estructura de métodos propuesta por Montilva y Barrios (2002), el método UML Component se puede estructurar en tres modelos: Modelo de producto que define la estructura y propiedades del producto a realizar, el modelo de proceso que detalla las actividades a realizar con la finalidad de elaborar el producto y el modelo de grupo, que describe las personas involucradas con su correspondiente rol en el desarrollo del proyecto.

3.1.1.1 Modelo de Producto del Método UML Component

A través de la Tabla 3.1 se visualiza el modelo de producto del método UML Component.

Tabla 3.1 El modelo de productos UML Component.

Producto	**Descripción**	**Artefactos**
Productos de requerimientos	Comprende los modelos: *conceptos de negocios*: donde se identifican las clases de entidades que existen y las asociadas; y *casos de uso* donde se muestran los tipos de usuarios y las funciones de interacción entre cada tipo de usuario y la aplicación.	1. Conceptos de negocio: Diagrama de Clases. 2. Casos de usos: Diagrama de casos de usos.
Productos de especificación	Describen los modelos: *tipo de negocio*, consiste en desplegar los tipos de entidades del negocio y sus detalles (atributos y restricciones, relevantes a la aplicación), es un refinamiento del modelo de conceptos de negocios, *Especificación de interfaz* representa una interfaz, en términos de: tipo de interfaz, información que incluye los tipos asociados a la interfaz y la especificación de cada operación: signatures, pre y postcondiciones), *especificaciones de componentes* y *arquitectura de componentes* donde se definen las interacciones entre componentes a través de sus interfaces.	1. Tipo de negocio: Diagrama de clases. 2. Especificación de interfaz: Diagrama de clases. 3. Especificaciones de componentes: Diagrama de clases. 4. Arquitectura de componentes: Diagrama de arquitectura de componentes.
Productos de software	Comprende los componentes de negocios y componentes del sistema y aplicación.	

3.1.1.2 Modelo de Proceso de UML Component

El modelo de proceso tiene como objetivos principales:

1. Entender los procesos de negocios que serán apoyados por la aplicación.

2. Especificar los requerimientos del sistema.

3. Identificar las interfaces de los componentes del negocio y del sistema, crear la arquitectura inicial de componentes y establecer las interacciones entre los componentes.

4. Elaborar las especificaciones de componentes, sus interfaces y sus restricciones.

3.1.2.3 Modelo de Actores de UML Component

El modelo de actores está principalmente enmarcado en:

1. *Especificador* (arquitecto).- persona que hace las especificaciones técnicas para un sistema o componentes pertenecientes a un sistema.

2. *Realizador*.- persona que construye un componente de acuerdo con una especificación de componente.

3. *Cliente*.- persona que escribe el software para usar un componente.

3.1.2 Método WATCH (Montilva & Barrios, 2003)

El método WATCH en la versión de Montilva & Barrios (2003) está referido al desarrollo de aplicaciones web. El método comprende el ciclo de vida de las aplicaciones web y se encuentra descrito en términos de tres componentes metodológicos:

1. *Modelo del producto***.-** Se refiere al producto que el método WATCH ayuda a generar.

2. *Modelo del proceso*.- Representación estructurada del conjunto de actividades que el grupo de desarrollo debe realizar con la finalidad de generar una aplicación empresarial.

3. *Modelo del grupo de desarrollo*.- Comprende actores, roles y organización del grupo de desarrollo.

3.1.2.1 Modelo de Producto del Método WATCH

El modelo de producto del método WATCH, se relaciona con la representación de los elementos comunes elaborados al final del uso del método.

El método WATCH, esta basado en la arquitectura de aplicaciones web de N capas, sin embargo en este caso comprende tres capas: Capa de presentación, capa de lógica de negocios y capa de datos, como se muestra en la Figura 3.2 – Modelo de producto del método WATCH.

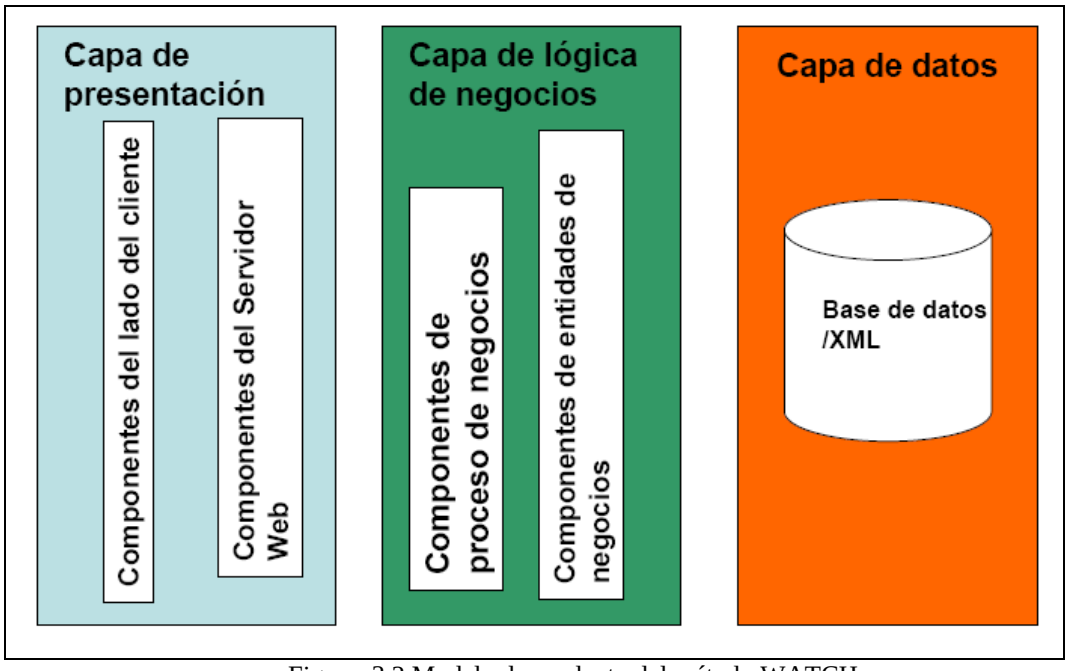

Figura: 3.2 Modelo de producto del método WATCH .
Fuente: Montilva & Barrios, (2002)

3.1.2.2 Modelo de Proceso del Método WATCH

Los procesos del método Watch, están divididos en procesos gerenciales y procesos de desarrollo.

3.1.2.2.1 Procesos Gerenciales

Tabla 3.2 Procesos Gerenciales del Método WATCH

Proceso	**Producto**
Gestión del proyecto	Descripción formal del proyecto a desarrollar. 1. Plan del Proyecto. 2. Estructura del grupo. 3. Mantener la coherencia y la comunicación entre los grupos. 4. Contrato del personal. 5. Documentación de las decisiones.
Gestión de la calidad del software	1. Plan de calidad del software. 2. Modelo de calidad.
Gestión de la configuración del software	1. Plan de cambio de configuraciones. 2. Reportes de las revisiones.
Verificación & Validación	1. Planes de Pruebas. 2. Resultados de las pruebas.
Gestión de riesgo	1. Lista de riesgos. 2. Probabilidades de cada uno de los riesgos y las técnicas de mitigación. 3. Recursos para la mitigación de riesgos. 4. Planes de mitigación de riesgos. 5. Planes de mitigación de riesgos aplicados.
Adiestramiento del personal del proyecto	1. Horario de adiestramiento, lista de personas a adiestrar. 2. Personal adiestrado.
Adiestramiento de usuarios	1. Horario de adiestramiento, lista de personas a adiestrar. 2. Personal adiestrado.
Documentación del sistema	Documentos del sistema

3.1.2.2.2 Procesos de Desarrollo

Tabla 3.3 Fases del proceso de desarrollo del Método WATCH

Fase	**Producto**
Modelado de negocios	1. Documento del Modelo del Dominio (DMD). 2. Documento del Modelo de Negocios (DMN). 3. Documento del Vocabulario del Dominio (DVD).
Definición y especificación de requerimientos	1. Documento de Especificación de Requerimientos Informal (DERI). 2. Documento de Especificación de Requerimientos Formales (DERF).
Diseño arquitectural de aplicación	1. Documento de Diseño de la arquitectura (DDA). 2. Documento de Diseño de la interfaz E/S (DDI). 3. Documento de Diseño Base Datos (DDBD).
Especificación de componentes	1. Documento de Plataforma de despliegue (DPD). 2. Documento de Especificación del componente (DEC). 3. Documento de Especificación de base de datos (DEBD).
Aprovisionamiento de los componentes	1. Interfaz Web: capa de presentación 2. Componentes. 3. Base de datos.
Ensamblaje de los componentes	1. Documento de Arquitectura Lógica Adaptada (DALA). 2. Aplicación ensamblada. 3. Documentos de las Pruebas de la Aplicación (DPA). 4. Documentos de la Aplicación Web (DAW).
Prueba de la aplicación web	1. Documento pruebas funcionales (DPF). 2. Documento pruebas comportamiento (DPC). 3. Documento pruebas de aceptación (DPA).
Entrega de la aplicación	1. Aplicación instalada en el ambiente de producción. 2. Adiestramiento del personal que va a utilizar la aplicación.

3.1.2.3 Modelo de Grupo del Método WATCH

El grupo de trabajo del método WATCH esta constituido por las personas que trabajan en los procesos gerenciales y las personas que trabajan en los procesos de desarrollo.

3.1.2.3.1 Procesos gerenciales: El líder del proyecto es el actor único de éstos procesos

3.1.2.3.1 Procesos de desarrollo: En la tabla 3.4 se detallan los actores que intervienen en el proceso de desarrollo del método WATCH.

Tabla 3.4 Actores del proceso de desarrollo del Método WATCH

Fase	**Actores**
Modelado de negocios	1. Analista de negocios. 2. Representante de los usuarios. 3. Líder del proyecto.
Definición y especificación de requerimientos	1. Analista de negocios. 2. Representante de los usuarios. 3. Líder del Proyecto. 4. Desarrollador de aplicación.
Diseño arquitectural aplicación	1. Desarrollador de aplicación. 2. Desarrollador de componentes de datos. 3. Desarrollador web.
Especificación de componentes	1. Desarrollador de componentes. 2. Desarrollador de componentes de datos.
Aprovisionamiento de los componentes	1. Desarrollador Web. 2. Desarrollador de componentes de negocios. 3. Desarrollador de componentes de datos.
Ensamblaje de los componentes	1. Desarrollador de data. 2. Desarrollador de componentes. 3. Desarrollador Web.
Prueba de la aplicación web	1. Desarrollador de data. 2. Desarrollador de componentes. 3. Desarrollador Web. 4. Representante de los usuarios.
Entrega de la aplicación	1. Líder del proyecto. 2. Desarrollador de la aplicación. 3. Usuarios o representantes de los usuarios.

3.1.3 WATCH – Component (Hamar,2003)

El WATCH – Component [Vanessa Hamar,2003] es un método que describe el ciclo de vida de un componente de software reutilizable. Detalla todo el proceso de especificación, hasta la liberación de un componente de software reutilizable. Este método complementa el Método WATCH Extendido en las fases de especificación y aprovisionamiento del componente, además puede ser utilizado de manera independiente, en la creación de componentes reutilizables o como complemento para los métodos de desarrollo basado en componentes. El método WATCH Extendido es derivado del método WATCH (Montilva and Barrios, 2003).

Este método fue diseñado a partir de la integración conceptual y metodológica de los métodos WATCH (Montilva and Barrios, 2003) y UML Component (Cheesman and Daniels, 2002). Comprende los tres modelos propuestos en el método WATCH: *modelo de producto*, donde se definen los componentes y su forma en cada una de las etapas propuestas por Cheesman & Daniels (2001), y la representación del contexto en general del componente. *Modelo de Proceso,* el cual define cada una de las actividades del desarrollo de componentes; y finalmente en el *modelo de grupo* se establecen los roles y las responsabilidades de cada uno de los actores que participan en el desarrollo de un componente.

3.1.3.1 Modelo de Producto del Método WATCH - Component

El modelo del producto, en el método WATCH – Component, es un componente de software reutilizable, que se plasma en las diferentes etapas por las que va a pasar el producto (el componente), las cuales son tomadas en cuenta durante la definición del modelo de procesos, para crear una relación entre el modelo de producto y el modelo de procesos.

Un componente puede existir en diferentes formas durante su ciclo de vida:

1. *Componente especificado*: Establece las características del componente y las funciones que realiza.

2. *Componente implementado*: Comprende la realización del componente.

3. *Componente instalado*: La instalación (despliegue) de la implementación del componente en una plataforma de ejecución determinada.

Mediante la Figura 3.3 se visualiza en el ciclo de vida de un componente de software reutilizable.

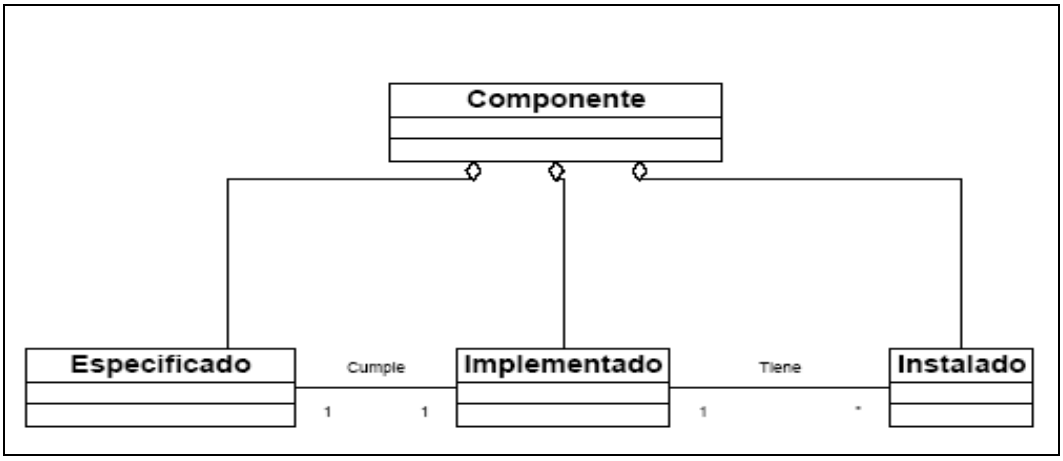

Figura: 3.3 Etapas en el ciclo de vida de un componente de software reutilizable.
Fuente: Hamar (2003)

3.1.3.2 Modelo de Proceso del Método WATCH - Component

El modelo de procesos de WATCH – Component está constituido por fases que comprende el ciclo de vida de un componente, desde su especificación hasta su inclusión en el repositorio. Las fases son: especificación del componente, aprovisionamiento del componente, prueba del componente, certificación del componente, liberación del componente y mantenimiento del componente. El método WATCH - Component hereda del método WATCH, los procesos gerenciales que son pilar fundamental del éxito del método WACTH.

El modelo de procesos de WATCH – Component comprende los procesos gerenciales y los procesos de desarrollo. Ver tabla 3.5 y 3.6 donde se describe los procesos y productos de los procesos gerenciales y las fases de los procesos de desarrollo.

Tabla 3.5 Procesos Gerenciales del Método WATCH – Component.

Proceso	**Productos**
Gestión del proyecto	1. Descripción informal del proyecto a desarrollar. 2. Plan del Proyecto. 3. Estructura del grupo de desarrollo. 4. Mantener la coherencia y la comunicación entre los grupos. 5. Grupo de Desarrollo. 6. Documentación de las decisiones tomadas.
Gestión de la calidad del software	1. Plan de calidad del software. 2. Modelo de calidad.
Gestión de la configuración del software	Plan de cambio de configuraciones.
Verificación & Validación	1. Planes de Pruebas. 2. Resultados de las revisiones.
Gestión de riesgo	1. Lista de riesgos. 2. Probabilidades de cada uno de los riesgos y las técnicas de mitigación. 3. Determinación de recursos para la mitigación de riesgos. 4. Planes de mitigación de riesgos. 5. Planes de mitigación de riesgos aplicados.
Documentación	Procesos documentados.
Adiestramiento	1. Horario de adiestramiento, selección de personas a entrenar. 2. Personal adiestrado. 3. Documentación y guías para el adiestramiento.

Tabla 3.6 Procesos de desarrollo del Método WATCH – Component.

Fases	**Producto**
Especificación del componente	1. Documento de especificación informal del componente (DEIC). 2. Meta-Arquitectura.. 3. Diagrama de interfaces. 4. Documento Especificación Formal del Componente (DEFC). 5. Documento del Contrato de uso. 6. Documento del Contrato de realización. 7. Documento de aceptación de la plataforma (DAP).
Aprovisionamiento del componente	1. Componente implementado. 2. Catálogos de Componentes (CC). 3. Catálogos de Sistemas Legados (CSL). 4. Catálogos de empresas externas (CEE). 5. Documento de resultados de la comparación (DRC). 6. Catálogos de Componentes Finales (CCF). 7. Patrones de comparación de componentes. 8. Resultados de la comparación de los componentes. 9. Decisión sobre la forma de aprovisionamiento. 10. Componente Adaptado. 11. Componente Implementado. 12. Sistema legado adaptado. 13. Componente provisto por terceros. 14. Documentación del componente seleccionado.
Prueba del componente	1. Documento de pruebas a realizar. 2. Documento pruebas funcionales. 3. Documento pruebas de comportamiento. 4. Documento pruebas de aceptación. 5. Documento del análisis de los resultados. 6. Documentos de las pruebas y resultados. Componente probado.
Certificación del componente	1. Componente certificado internamente (DCI). 2. Componente certificado por terceras partes (DCE).
Liberación del componente	1. Documento de catalogación del componente. 2. Documento de liberación del componente. 3. Documento de publicación del componente.

3.1.3.3 Modelo de Grupo del Método WATCH - Component

Los actores y sus roles de método WATCH - Component son descritos en la tabla 3.7.

Tabla 3.7 Actores del Método WATCH – Component(a).

ROL	RESPONSABILIDAD
Líder del Proyecto	Responsable del proyecto.
Arquitecto de Componentes	Responsable de la planeación de la arquitectura, de la infraestructura técnica de los componentes cubriendo áreas como: interfaces, interacciones entre los componentes, manejo de la data a almacenar, manejo de estándares del *middleware*.
Asesor de Componentes	Actúa como el centro del aprovisionamiento de componentes, debe conocer las estrategias del mercado, monitorear la industria de los componentes para evaluar las oportunidades de reutilización.
Gerente de Reutilización	Planea y controla las actividades del proyecto en cuanto a las decisiones de reutilización.
Gerente de aprovisionamiento	Es el responsable de la selección, adaptación, y control de componentes que provienen de fuentes externas.
Diseñador de Componentes	Modela los requerimientos genéricos, desarrolla las especificaciones de los componentes y asegura la integración con otros componentes.
Experto en sistemas legados	El experto en sistemas legados da consejos sobre las aplicaciones como bases de datos, paquetes de software que pueden ser envueltos o adaptados.
Desarrollador de Componentes	El desarrollador de componentes está encargado de la implementación de los componentes que no hayan sido encontrados mediante alguna otra forma de aprovisionamiento, es decir, desde cero.
Certificador de componentes	El componente debe ser certificado en cuanto a una serie de aspectos que deben ser cumplidos en su totalidad. Esta persona debe inmiscuirse en la parte de desarrollo y verificar los productos.
Realizador de pruebas de componentes	Este rol se encarga de lo relacionado con las estrategias de pruebas, cuando deben extenderse, especializarse o modificarse las interfaces, también debe poder comparar los resultados obtenidos en las pruebas con los deseados.

Fuente: Hamar (2003).

Tabla 3.7 Actores del Método WATCH – Component(b).

ROL	RESPONSABILIDAD
Administrador del Repositorio de Componentes	En el momento cuando el componente es liberado, este es colocado dentro de uno o más repositorios, cada uno de estos repositorios es gerenciado por un administrador de repositorios.

Fuente: Hamar (2003).

3.1.4 Arquitectura de software - Modelo de Vista 4+1

La arquitectura de software es un conjunto organizado de elementos, los cuales son utilizados en el proceso de toma de decisiones, al tiempo que proporciona definiciones y lenguaje de la estructura y funcionalidad del sistema, de manera que exista comunicación entre los participantes del proyecto. La arquitectura de software se vale de la construcción de abstracciones convirtiéndolas en diagramas definidos.

Para la representación de los diagramas no existen estándares, sin embargo uno de los modelos más conocidos es el creado por Philippe Kruchten llamado "*modelo de vista 4+1*" , este modelo define 5 vistas diferentes (Kruchten, 2005):

1. *Vista lógica*: Describe el modelo de objetos a desarrollar, para lo cual son utilizados los diagramas de clases.

2. *Vista de proceso*: Muestra la concurrencia y sincronización de los procesos, reflejando la organización de módulos de software dentro del entorno de desarrollo.

3. *Vista física*: Donde visualiza la ubicación del software en el hardware, esta vista se centraliza en la estructura de los componentes ejecutables del sistema,

4. *Vista de desarrollo*: Trata sobre la organización del entorno de desarrollo, de manera que el equipo de desarrollo comprenda mejor la topología de un sistema distribuido.

5. *Vista de certificación*: Validación de las cuatro vistas anteriores con la funcionalidad requerida del sistema, para lo cual puede utilizar diagramas de casos de uso, especificaciones de casos de uso, diagrama de escenarios, diagrama de actividad y diagrama de estados.

3.2 Estructura del método

El diseño del método se basa en definiciones de la Ingeniería de Métodos (ME) y de la Ingeniería del Software (SE). Fundándose en el principio de ME, que expone, que un método bien definido se debe describir en términos de tres modelos: Modelo de producto, modelo de proceso y modelo del equipo de desarrollo.

Para la elaboración del *modelo de producto*, se utilizaron las etapas propuestas por Cheesman & Daniels (2001), y el modelo de vista 4+1 para representar el contexto en general del SW.

En relación al *modelo de proceso*, se sigue el modelo de proceso del método WATCH – Component de Vanessa Hamar (2003) y para la especificación del diagrama de actividades se utilizo UML Components de Cheesman and Daniels(2001).

En el *modelo de grupo* se describen las funciones y responsabilidades de cada una de las personas que participan en el desarrollo de un SW.

3.3 Características del método

1. Método para desarrollar SW: DESWeb es un método que está orientado bajo una arquitectura orientada a servicios (SOA - *Service Oriented Architecture)* arquitectura distribuida de N capas. Las capas de su arquitectura son: usuario, presentación, lógica del negocio y datos.

2. Comprende tres modelos: Modelo de producto, modelo de proceso y modelo de grupo. Acogiéndose a los principios de ME y a la extensión que hace el método WATCH.

3. Comprende todo el ciclo de desarrollo de un SW: La constitución de las fases comprende el ciclo de vida de un SW, desde su especificación hasta su registro en el repositorio.

4. Planificación y control durante el desarrollo de un SW: Al igual que el método WATCH y el WACTH – Component, el método DESWeb contempla los procesos gerenciales que permiten tener un control del proyecto desde sus inicios.

5. Comunicación efectiva entre los actores participantes: Dentro de las funciones del líder del proyecto está contemplado que debe llevar un control, tanto de las actividades de cada fase de desarrollo, como la supervisión del personal, garantiza que dentro del grupo de desarrollo exista un ambiente de trabajo agradable para todos sus integrantes.

CAPÍTULO IV

EL MODELO DE PRODUCTOS DEL MÉTODO DESWEB

Este capítulo describe el modelo del producto del método DESWeb. El modelado se describe a través del producto que consiste en un SW y las diferentes etapas que pasa un software de esta naturaleza.

Está estructurado en cuatro (4) secciones: La Sección 4.1 contempla las vistas de un SW donde se menciona el concepto de Arquitectura de Software (AS) y las vistas que permitirán representar un SW.

La Sección 4.2 está relacionada con la vista funcional o lógica que describe las diferentes etapas del ciclo de vida de un SW, la Sección 4.3 trata de la vista estructural o de los procesos que permiten crear una categorización de los SW, de acuerdo con sus estados, y la última Sección 4.4, está relacionada con la vista dinámica que presentan todas las posibles fases o procesos que pueda recorrer un SW.

4.1 Vistas de un SW

La AS es la parte de la Ingeniería de Software que se encarga de la descripción y la estructura de un sistema como un conjunto de componentes, para establecer adecuadamente los distintos subsistemas, y poder integrar los diferentes grupos de desarrollo en el proyecto.

Mediante la AS se plasman varias perspectivas de un sistema, programa o aplicación, las cuales pueden ser mostradas a través de modelos o vistas. Donde cada vista o modelo representa una descripción parcial de una misma arquitectura.

Las vistas forman parte del único aspecto obligatorio en la práctica recomendada RP-1471 sobre Arquitectura de Software de la IEEE(Hilliard, 2001). Uno de los modelos de vistas más conocidos es el modelo de vistas 4+1 presentado por Philippe Kruchten, el cual está constituido de 5 vistas:

1. Vista Lógica
2. Vista de proceso
3. Vista física
4. Vista de desarrollo
5. Vista de certificación

Para el modelo del producto del método DESWeb, se presenta en función de tres vistas diferentes: Vista funcional o lógica, vista estructural o de procesos y una vista dinámica o de desarrollo, donde cada una de ellas representa parte de un SW.

4.2 Vista Funcional de un SW

Un SW pasa por diferentes etapas en su ciclo de vida: (1) SW *Especificado:* cuando se establece las características y funciones. (2): SW *Implementado*: relacionado con construcción de SW que incluye el desarrollo, prueba de la implementación, definición de la descripción de la interfaz y la definición de la descripción de la implementación. (3) *SW Publicado:* referido al registro del SW en un registro UDDI, (4) SW *Desplegado*: consiste en la publicación de la interfaz dentro de un ambiente de ejecución (servidor de aplicaciones web), para luego publicarlo de manera que quede en funcionamiento disponible para utilizarlo por el cliente del servicio que puede realizar búsqueda y efectuar operaciones, (5) *SW Contratado:* proceso concerniente a la figura de un contrato existente entre el proveedor del servicio y el cliente, (6) *SW Usado*: proceso que se realiza cuando los clientes invocan un SW que ya ha sido publicado y (7) *SW Retirado:* se presenta

cuando el proveedor del servicio decide no seguir ofreciendo un SW determinado. Esta vista es mostrada a través de la Figura 4.1, utilizando un diagrama de clases UML.

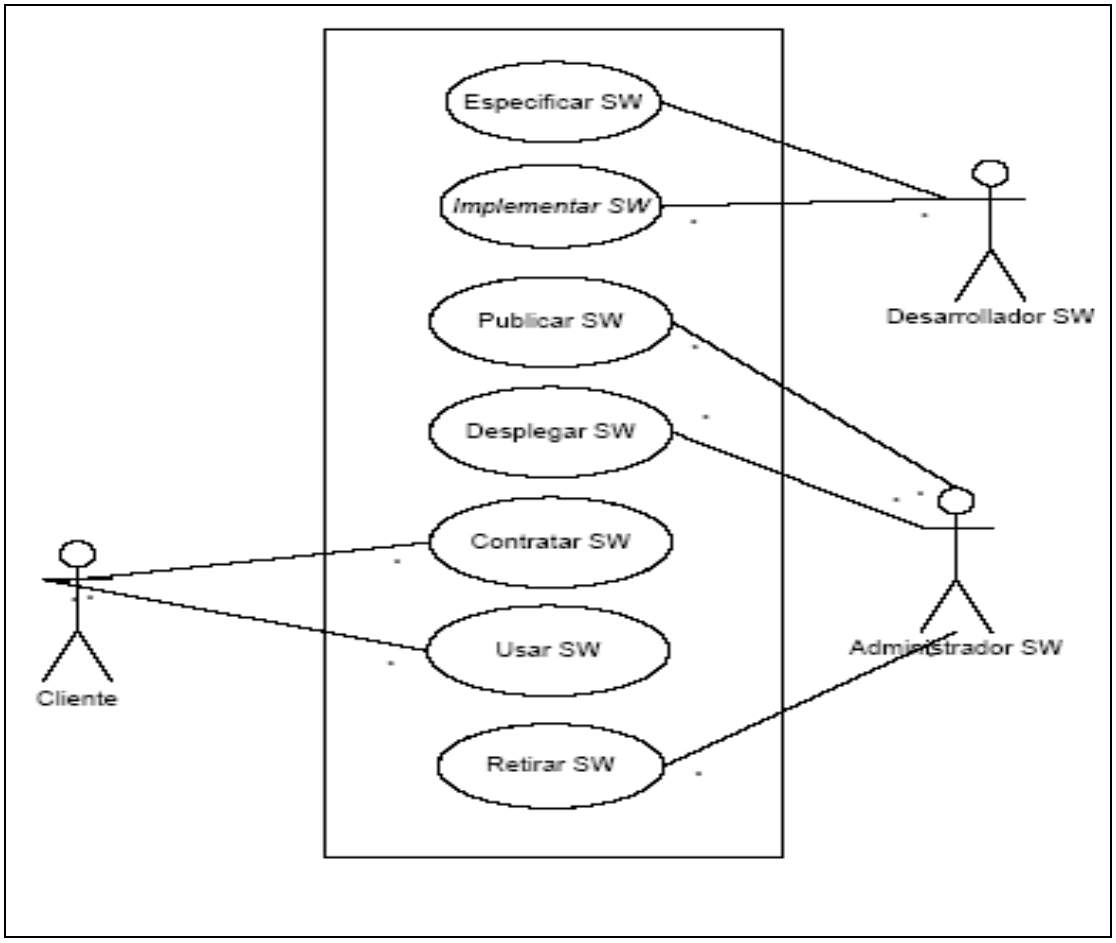

Figura 4.1 Vista funcional de un SW.

4.3 Vista Estructural de un SW

Tomando la propuesta de Cheesman & Daniels (2001) en relación con la presentación del modelo de un producto, que permite crear una categorización de los SW de acuerdo con sus estados, esto es mostrado en la Figura 4.2 mediante un Diagrama de Clases.

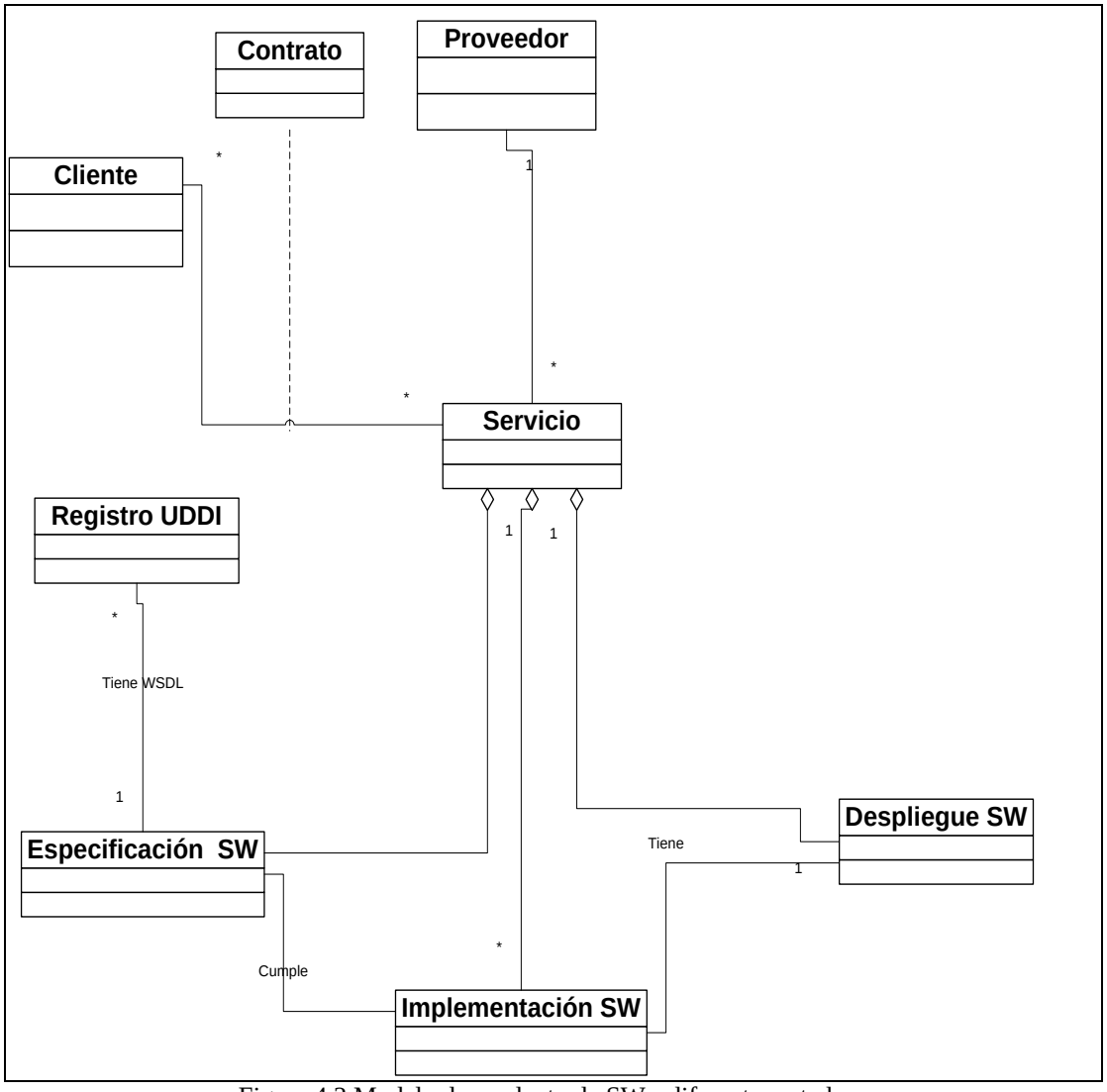

Figura 4.2 Modelo de producto de SW - diferentes estados.

4.4 Vista Dinámica de un SW

La dinámica de un sistema está establecida por:

1. Todas las posibles fases que sus objetos pueden tener.
2. Todas las posibles sucesiones de acontecimientos que pueden suceder.
3. Todas las actividades posibles, de una fase a otra, como consecuencia de acontecimientos que afectan a los objetos.

A través de la Figura 4.3 - Diagrama de estados de un SW - se muestra el comportamiento de un SW durante su ciclo de vida, donde los aspectos a resaltar son los siguientes:

1. Etapa inicial de Especificación del SW.
2. Seguidamente está la fase de Implementación basada en las especificaciones del SW.
3. La Fase de Despliegue que puede estar a la par de la fase de Contratación.
4. Para que se lleve a cabo la fase de Contratación de SW, el SW debe estar publicado. La publicación puede ser realizada en repositorio UDDI interno o externo o ambos.
5. Un SW será utilizado siempre y cuando se hayan cumplido las Fases de despliegue o contratación.

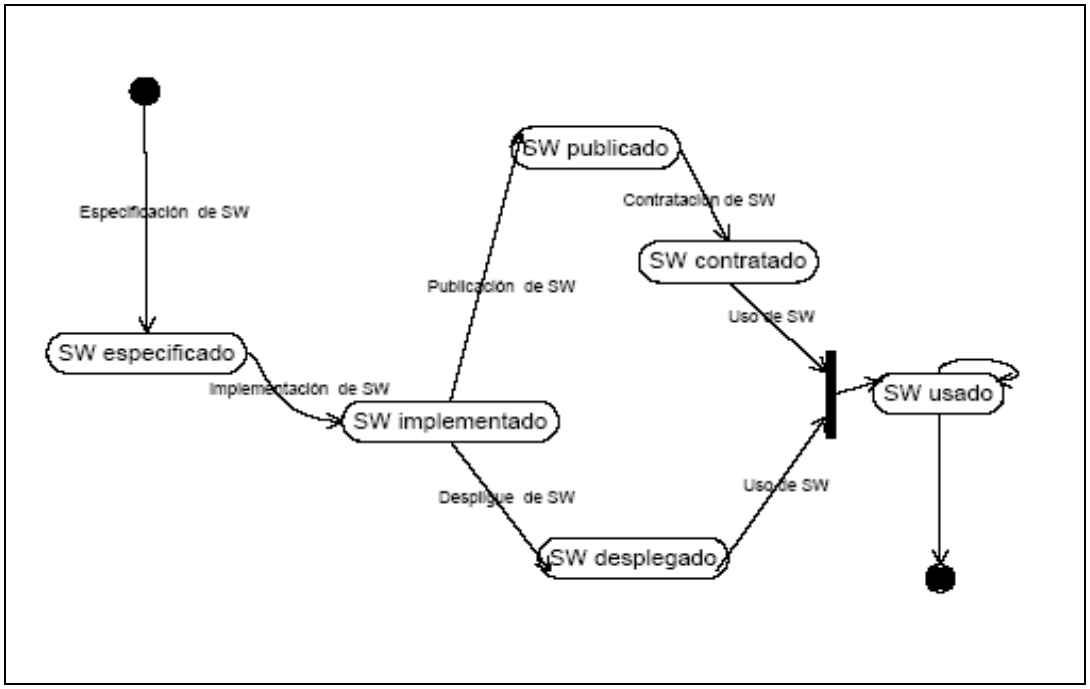

Figura 4.3 Diagrama de estados de un SW.

4.4.1 Modelo de un SW Especificado

Mediante el modelo de una especificación de un SW se pueden determinar aspectos principales en el proceso de la especificación del mismo, para obtener los conceptos asociados a su forma, que pueden ser considerados como productos. La Figura 4.4 muestra la interfaz de un componente SW

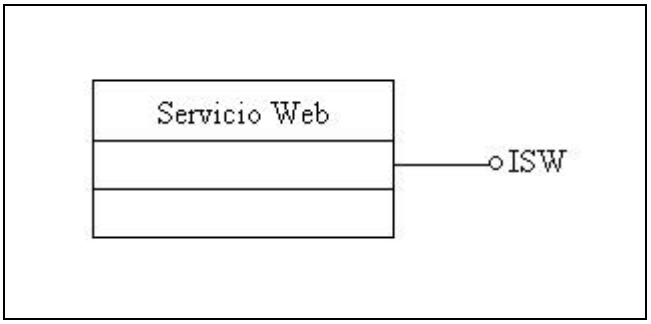

Figura 4.4 Interfaz de un componente SW.

Según Cheesman & Daniels (2001), la especificación viene dada por las descripciones de las *interfaces ofrecidas* (mediante las operaciones: signaturas, definiciones, entradas, salidas, restricciones y condiciones para aplicarse y el modelo de información) y la *realización*; cada una de ellas hacen uso de contratos establecidos. Tomando textualmente esta definición, la Figura 4.5 muestra el modelo de la especificación de un SW.

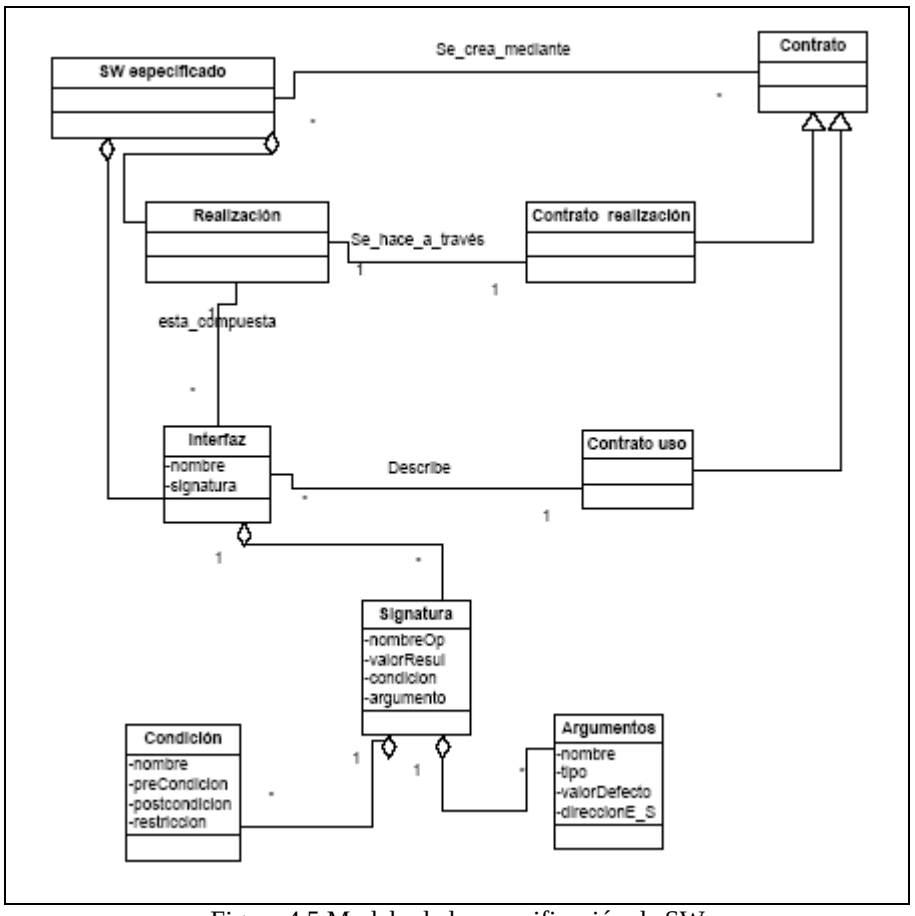

Figura 4.5 Modelo de la especificación de SW.

En el modelado de un SW especificado - Figura 5 - muestra las siguientes representaciones:

1. La especificación de un *SW* se crea a través de contratos.

2. Un SW se definen mediante la especificación de la *realización y las interfaces.*

3. La especificación de la *realización* de un SW está compuesta por la especificación de cada una de sus *interfaces.*

4. *El contrato de realización* se hace según lo especificado en la *realización* del SW.

5. *El contrato de uso* cumple con lo especificado en las *interfaces* del SW.

6. Las interfaces se definen mediante la especificación de las *signaturas.*

7. Las signaturas se definen mediante la especificación de los argumentos y condición.

4.4.2 Modelo de un SW Implementado

El modelo de un SW implementado es el resultado de los procesos de diseño, codificación y prueba del SW especificado.

En la implementación, es necesario considerar la especificación de la interfaz, la cual involucra varios conceptos: de acuerdo con Cheesman & Daniels (2001) y Han (1999).

Igualmente, Cheesman & Daniels (2001) consideran que la especificación de interfaces debe establecerse las operaciones provistas, que incluyen signaturas y definiciones. Donde cada operación es relacionada con un contrato de grano fino y es descrita por:

1. *Precondiciones*: exposición de las situaciones donde las post-condiciones pueden ser utilizadas.

2. *Post-condiciones*: representación de las derivaciones de las operaciones con sus parámetros y modelo de información.

Las especificaciones de las operaciones incluyen:

1. *Parámetros de entrada*: especifica la información provista o pasada por el SW.

2. *Parámetros de salida*: especifica la información actualizada o devuelta por el SW.

3. Otros *cambios de estado* resultante del SW.

4. Otras *restricciones* que se apliquen.

Según Han (1999), al especificar la interfaz se debe incluir los siguientes elementos:

1. *Signaturas:* concepto indispensable, dado que describe la funcionalidad, la cual está constituida por las propiedades, operaciones y eventos del objeto.

2. *Modelo de información*: definición abstracta de cualquier información o estado que es definida entre las peticiones del cliente y la interfaz, y otras restricciones de esta información.

La Figura 4.6 muestra el modelo de un SW implementado, donde en los aspectos considerados se tomó en cuenta las definiciones anteriores:

1. Un SW implementado es el resultado de diseñar, codificar y probar, en una plataforma de desarrollo, un SW especificado.

2. Un SW implementado sigue las especificaciones de un contrato.

3. Un SW implementado está formado por la implementación de las interfaces y la implementación interna del SW.

4. Mediante la interfaz se comunica la aplicación con la implementación interna del SW.

5. Las interfaces de un SW implementado pueden ser del tipo provistas o requeridas.

6. Las interfaces proveen servicios.

7. Las interfaces implementadas se rigen por lo señalado por el contrato de uso.

8. Una interfaz puede estar compuesta por una o más operaciones.

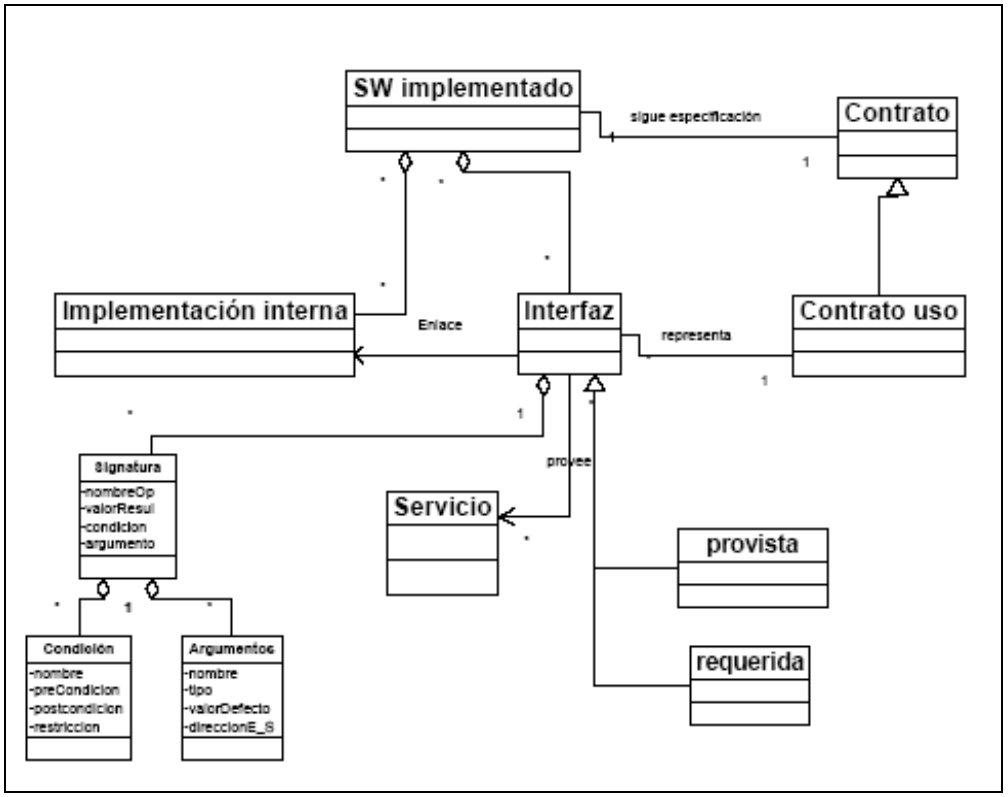

Figura 4.6 Modelo de un SW implementado.

4.4.3 Modelo de SW Desplegado

El modelo de SW desplegado está relacionado con el despliegue del SW en un registro UDDI interno en el ambiente de ejecución, con el propósito que sea invocado internamente dentro de la organización, ver Figura 4.7.

En este modelado son resaltados ciertos puntos:

1. El SW es registrado en un UDDI interno (ambiente de producción)

2. El SW puede ser desplegado en una plataforma de aplicaciones empresariales, de las características de .NET, J2EE, entre otras.

3. Las solicitudes de las interfaces dependerán del tipo (Stub, Proxy, DII - Interfaz de invocación dinámica - *Dynamic Invocation Interface*).

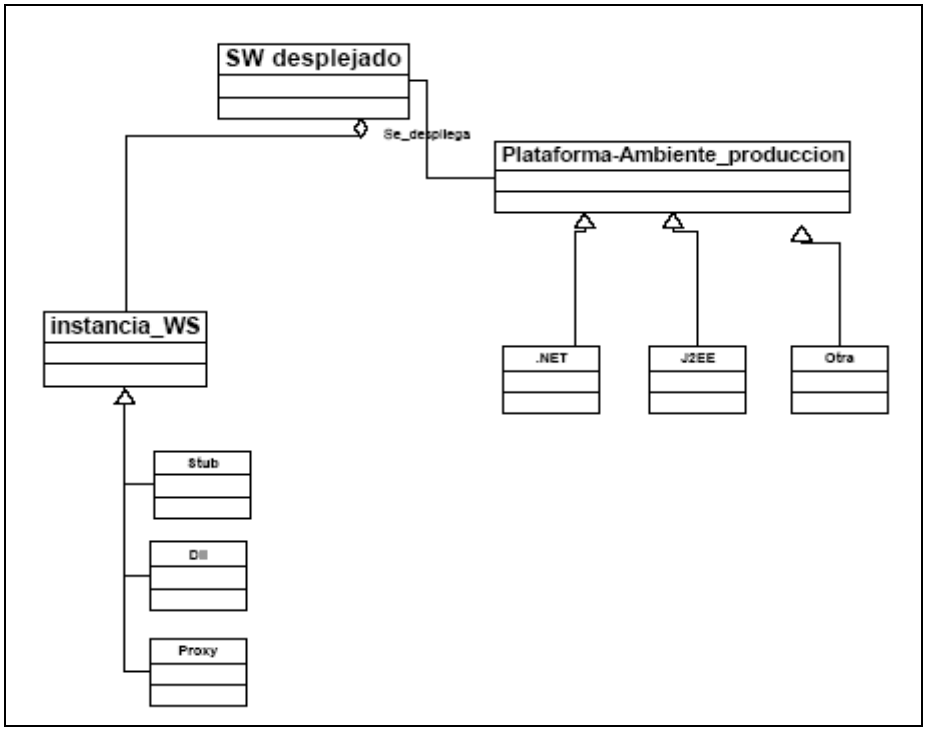

Figura 4.7 Modelo de un SW desplegado.

4.4.4 Modelo de SW Publicado

El modelo de SW publicado se refiere al registro de un servicio en UDDI con la finalidad que sea invocado por cualquiera que desee hacer uso del mismo. Este proceso se repite cada vez que es publicado un SW en este ambiente.

Este modelo es mostrado a través de la Figura 4.8 y los aspectos a resaltar son los siguientes:

1. Un SW publicado pertenece a un proveedor
2. Un SW publicado está desplegado en una plataforma de ejecución.
3. Un proveedor tiene varios SW
4. Un proveedor de SW registra SW en uno o varios registros UDDI
5. Un cliente busca SW de su interés en el registro UDDI.

6. Un cliente invoca SW de acuerdo al resultado (WSDL) retornado.

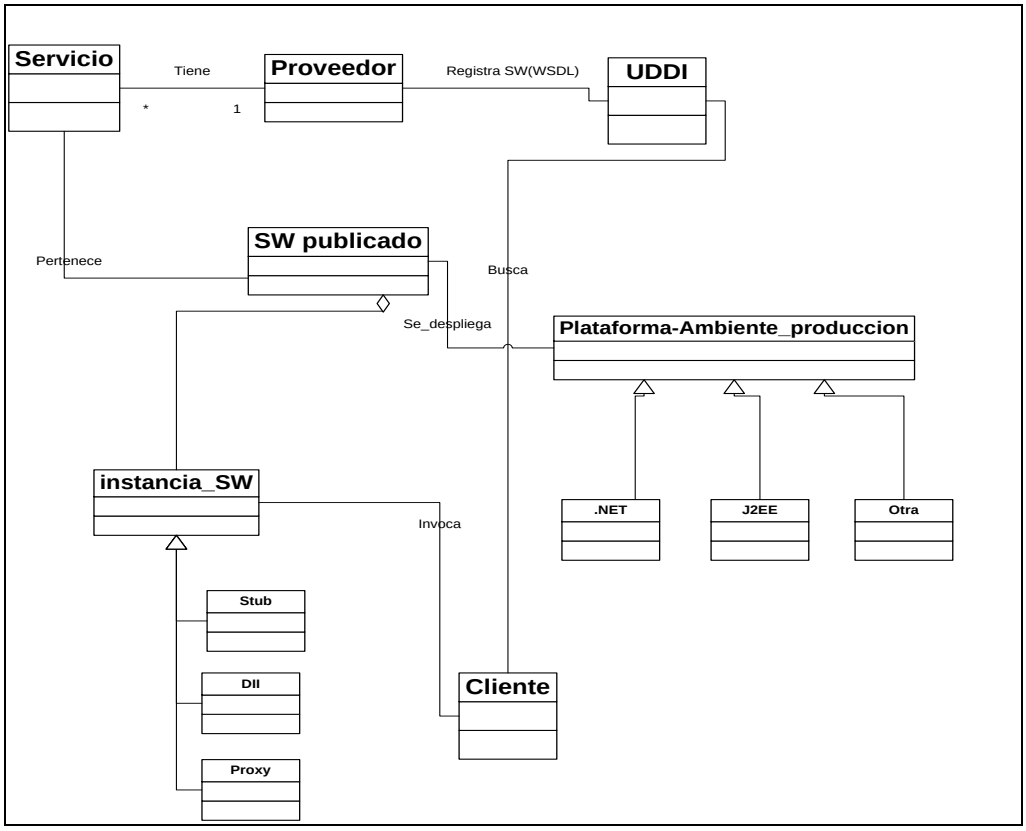

Figura 4.8 Modelo de un SW Publicado.

4.4.5 Modelo de SW Contratado-Usado

El modelo de SW contratado está relacionado con las operaciones realizadas por un cliente interesado en uno o varios SW y el proveedor del SW, con la finalidad de establecer condiciones y pautas del uso del SW.

La Figura 4.9 muestra este modelo. Dentro de los aspectos a resaltar están:

1. Un SW contratado es un servicio que ha sido desplegado y publicado.

2. Existe un contrato de cliente entre el proveedor del servicio y el cliente que requiere el acceso al servicio.

3. El contrato de cliente, contempla cláusulas que deben ser cumplidas.

4. El cliente del SW, puede hacer uso del servicio siempre y cuando previamente se haya celebrado un contrato de cliente.

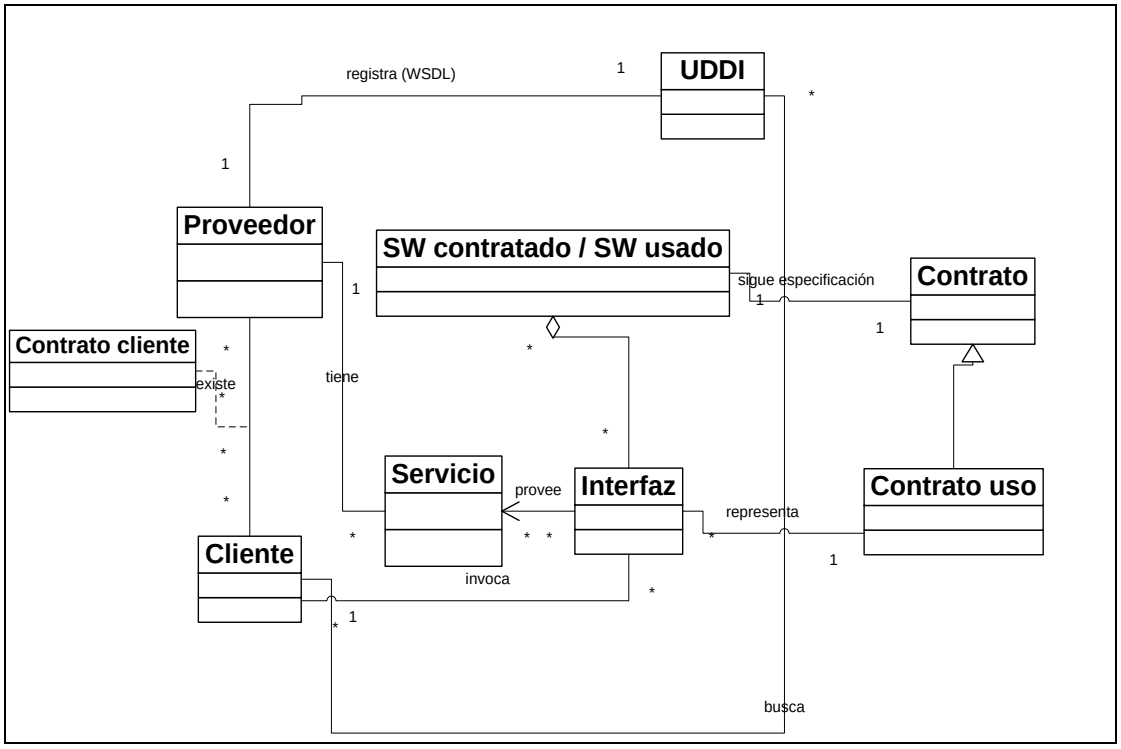

Figura 4.9 Modelo de un SW Contratado.

CAPÍTULO V

EL MODELO DE PROCESOS DEL MÉTODO DESWEB

En este capítulo se describe el modelo de procesos del método DESWeb. El modelado se describe a través de los diferentes procesos por los que pasa el producto, el cual consiste en un SW.

Cada proceso está constituido por: una definición general del mismo, los productos generados y el flujo de trabajo, que es representado mediante un diagrama de actividades conjuntamente con una tabla descriptiva, la cual detalla las tareas que el grupo de desarrollo del SW debe ejecutar.

Este capítulo está integrado por tres secciones, la sección 5.1 trata del modelo de procesos, la sección 5.2 se describen las entradas y salidas del proceso de desarrollo de un SW, la sección 5.3 se especifican los procesos gerenciales que conforman parte de los procesos del método DESWeb y la sección 5.4 se describen los procesos técnicos del método DESWeb.

5.1 Modelo de procesos

El modelo de procesos del método DESWeb se basa en los métodos WATCH y WATCH Component. Los procesos están constituidos por procesos gerenciales y procesos técnicos. Los procesos técnicos se van ejecutando de manera semejante al movimiento de las agujas del reloj, con la propiedad de poder avanzar al próximo proceso o retroceder al anterior de acuerdo a los resultados obtenidos en el proceso gerencial denominado Verificación y Validación y/o a la toma de decisión del encargado del proyecto.

Los procesos comprendidos en el método DESWeb engloban el ciclo de vida de un SW: se inicia con la especificación del SW hasta su publicación en un registro UDDI, tal como es mostrado en la Figura 5.1

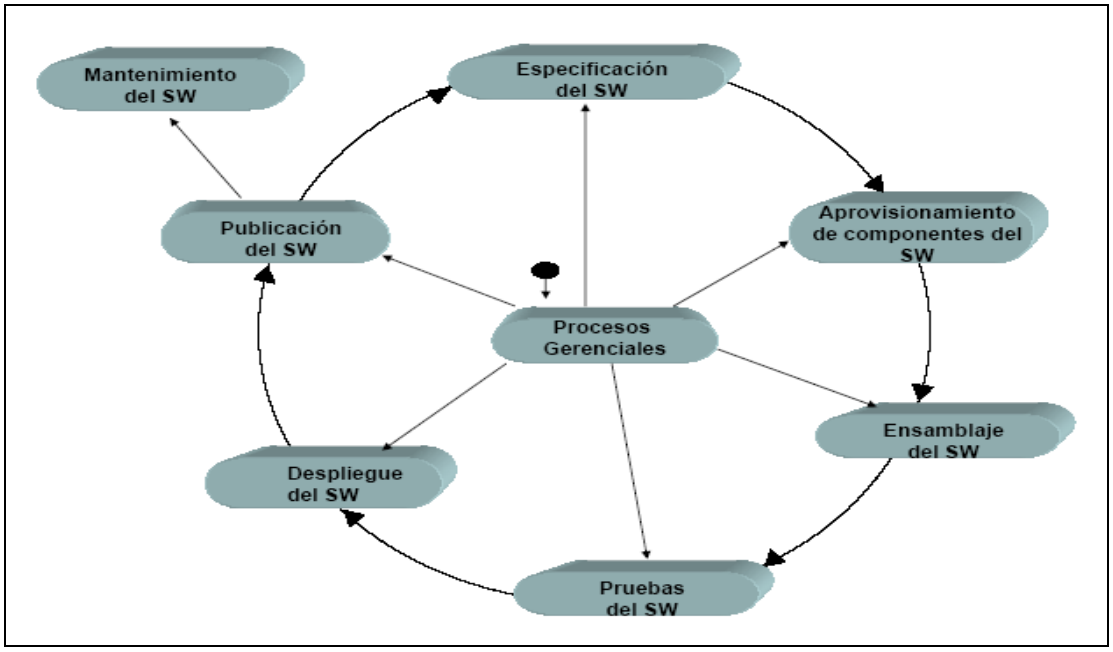

Figura 5.1 Modelo de procesos DESWeb.

Los procesos del Método DESWeb pueden ser vistos, también, como una cadena de valor, como se muestra en la Figura 5.2. Los procesos de desarrollo son apoyados por los procesos gerenciales.

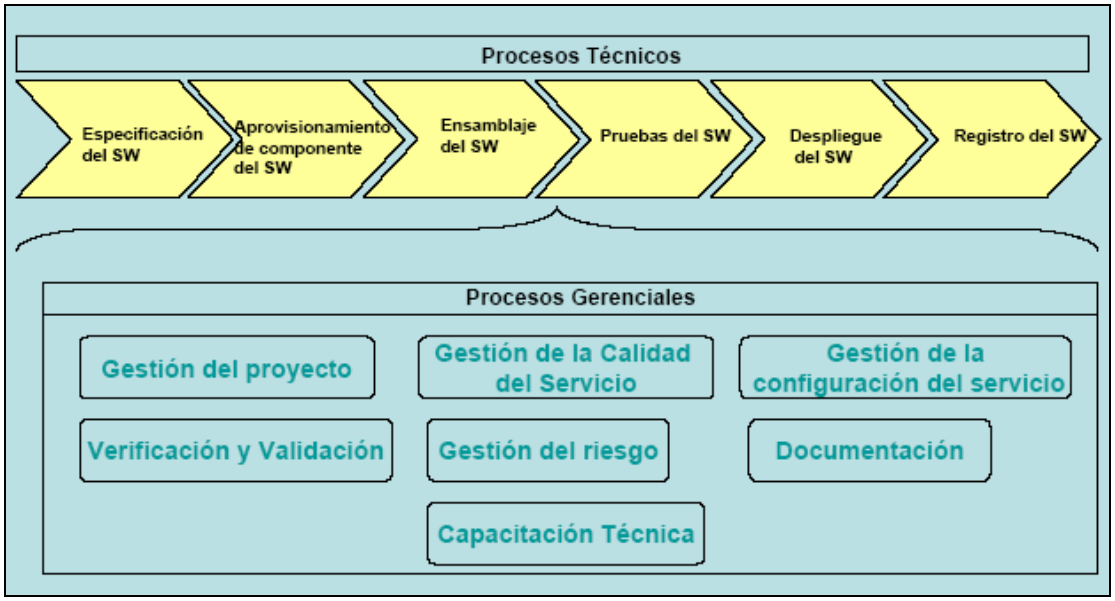

Figura 5.2 Modelo de procesos DESWeb.

5.2 Entrada/Salida de un Proceso de Desarrollo de SW

El proceso de desarrollo de SW contempla unas entradas y salidas; entradas que sirven de insumo al desarrollo del SW, y generan salidas que son el producto del proceso de desarrollo. La figura 5.3 muestra tales insumos y salidas de un proceso de desarrollo de un SW.

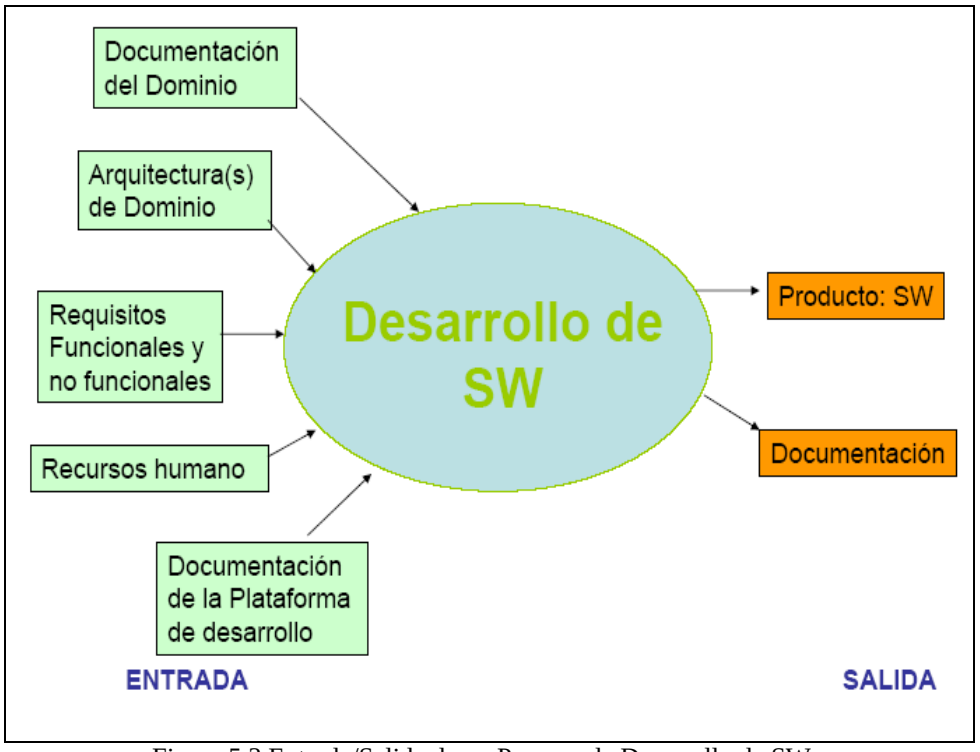

Figura 5.3 Entrada/Salida de un Proceso de Desarrollo de SW.

Considerando que hay elementos claves en la especificación de requisitos del proceso de desarrollo de SW y que deben ser tomados en cuenta; uno de estos elementos es lo relacionado a la Seguridad, es por esto que hay que hacer énfasis en el aseguramiento de la seguridad como una actividad preventiva.

Una de las recomendaciones para el proceso de desarrollo de SW, es colocar el proceso de seguridad como un proceso indispensable; Y es aquí donde se puede hacer uso de la Programación Orientada a Aspectos, conocida en el idioma inglés como AOP: Aspect Oriented Programming.

Wikipedia (2006), considera que la *Programación Orientada a Aspectos* (POA) es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos.

Los objetivos de la POA es separar conceptos para que cada decisión se tome en un lugar concreto y no disgregado por la aplicación; y minimizar las dependencias entre ellos con el fin de desarticular los distintos elementos que intervienen en un programa.

Considerando que un aspecto es la unidad básica de la POA, y pueden definirse como las partes de una aplicación que describen las cuestiones claves relacionadas con la semántica esencial o el rendimiento. La seguridad vendría hacer un aspecto a considerar en el diseño y la implementación del desarrollo de un SW.

Otra recomendación, en el uso del método DESWeb, es hacer uso de herramientas CASE (Computer Arded Software Engineering, Ingeniería de Software Asistida por Computador) en las diferentes fases del método DESWeb, con la finalidad de aumentar la productividad en el desarrollo de los SW, de manera de reducir costos en términos de tiempo y dinero.

5.3 Procesos Gerenciales

Los procesos gerenciales contienen una serie de actividades y tareas, que deben ser llevadas a cabo por el encargado del proyecto durante todo el ciclo de desarrollo del SW.

Es importante resaltar la importancia que tiene la gerencia de proyectos de software, dado que es un proceso cuyo objetivo es producir, dentro de las restricciones de costo y tiempo, un producto de *software* que satisfaga los requisitos impuestos por sus usuarios. Los aspectos claves de la gestión de un proyecto están en:

1. *Problema:* comunicación con el dueño del problema – solución adecuada. consiste en la definición del proyecto, objetivos del proyecto y el ámbito. Igualmente debe contener el plan del proyecto.
2. *Proceso:* métodos, técnicas, herramientas eficaces. Buscar el mejor enfoque considerando: limitaciones impuestas por fechas limites, tipo de problema, restricciones de presupuesto, disponibilidad de personal.
3. *Personal***:** esfuerzo humano intenso. consiste en la búsqueda y retención de personal con talento necesario para desarrollar proyectos de software. Mediante reclutamiento, selección, entrenamiento, retribución, entre otros

El Método DESWeb sigue la propuesta del método WATCH y método WATCH Component en relación a los procesos gerenciales, muchos de los cuales son propuestos por el estándar IEEE 1074 [IEEE95] para la elaboración de modelos de procesos de software. Estos procesos están constituidos por actividades, tareas, técnicas propuestas para las actividades y los productos obtenidos en cada proceso.

A continuación, se describe cada uno de los procesos gerenciales.

5.3.1 Gestión del Proyecto

Este subproceso muestra el inicio del proyecto, así como la planificación que debe contener el día a día de la ejecución del proyecto. Igualmente contempla la organización, dirección y administración del grupo de desarrollo del SW, junto con el control del proyecto. Las actividades y productos de este subproceso es visualizada en la tabla 5.1.

Tabla 5.1 Gestión del Proyecto - Procesos Gerenciales (a).

Gestión del Proyecto			
Actividad	**Tareas**	**Técnica**	**Producto**
Iniciación del proyecto	1. Establecer los objetivos y el propósito del proyecto y sus requisitos más generales.	Diagramas Pert-CPM	Informe inicial del proyecto.

Tabla 5.1 Gestión del Proyecto - Procesos Gerenciales (b).

Gestión del Proyecto			
Actividad	**Tareas**	**Técnica**	**Producto**
	2. Desarrollar las estimaciones iniciales del proyecto para cubrir los requisitos generales: tamaño, esfuerzo, tiempo, costos, horarios y recursos. 3. Determinar las actividades necesarias y las más relevantes en cada una de las fases. 4. Documentar la iniciación del proyecto.		
Planeación del proyecto	1. Desarrollar la estructura del trabajo. 2. Establecer los cronogramas de actividades. 3. Desarrollar las estimaciones del proyecto. 4. Identificar los estándares del proyecto. 5. Establecer los compromisos del proyecto. Documentar la planeación del proyecto.	Diagramas Pert-CPM	Informe del Plan del Proyecto.
Organización del grupo de desarrollo	1. Definición del grupo de desarrollo de acuerdo al Modelo de Actores del método. 2. Asegurar que el grupo de desarrollo entiendan el trabajo asignado.	Organigramas	Estructura del grupo de desarrollo.
Dirección del grupo de desarrollo del proyecto	1. Asegurar la interacción entre los miembros de los equipos 2. Asegurar que el grupo de desarrollo tenga una visión total del proyecto. 3. Establecer los métodos de interacción de los diferentes Actores. 4. Identificar los problemas que puedan afectar a los Actores y su interacción. 5. Documentar las tareas y la forma de comunicación entre los Actores.	Técnicas de Liderazgo Reuniones Dinámicas de grupo	Informe contentivo de las tareas del grupo de trabajo y la organización de la infraestructura de comunicación.

Tabla 5.1 Gestión del Proyecto - Procesos Gerenciales (c).

Gestión del Proyecto			
Actividad	**Tareas**	**Técnica**	**Producto**
Administración del grupo de desarrollo	1. Selección del personal. 2. Contratación del personal 3. Evaluación del personal 4. Retiro del personal	Técnicas de gestión de RRHH	Contratos
Control del proyecto	1. Gestión de los recursos del proyecto. 2. Toma de decisiones y acciones correctivas 3. Supervisión de los avances del proyecto. 4. Supervisiones técnicas. 5. Actualización del plan del proyecto. 6. Documentación de las decisiones tomadas y justificaciones.	PERT/CPM Monitoreo de proyectos Toma de decisiones	Plan del Proyecto Actualizado

5.3.2 Gestión de la Calidad del Servicio

Consiste en la planificación de actividades necesarias que aporten confianza en el producto resultante y éste en correspondencia con los requisitos funcionales, de manera de mantener bajo control los procesos y eliminar las posibles causas de defectos durante el ciclo de vida del SW. Las actividades y productos de este subproceso es mostrada en la tabla 5.2.

Tabla 5.2 Gestión de la Calidad del Servicio - Procesos Gerenciales(a)

Gestión de la Calidad del Servicio			
Actividad	**Tareas**	**Técnica**	**Producto**
Planificación de la calidad del servicio	1. Determinar cada uno de los procedimientos y estándares de calidad del servicio. 2. Definir las métricas de calidad que permitan medir los resultados de las metas de calidad que se desean alcanzar. 3. Identificar actividades que permitan mejorar la calidad del proyecto. 4. Documentar - plan de calidad.	ISO 1074, SPICE, Modelo McCall y Modelo FURPS.	Informe contentivo del Plan de Calidad

Tabla 5.2 Gestión de la Calidad del Servicio - Procesos Gerenciales (b)

Gestión de la Calidad del Servicio			
Actividad	**Tareas**	**Técnica**	**Producto**
Aseguramiento de la calidad del software	1. Asegurar que los procedimientos y estándares de calidad se cumplan. 2. Elaborar las métricas de calidad 3. Auditar los productos elaborados. 4. Tomar acciones correctivas. 5. Elaborar informe con los resultados.	Patrones de comparación de resultados	Informe de resultados del aseguramiento de la calidad del software.

5.3.3 Gestión de la Configuración del Servicio(GCS)

Proceso que identifica, organiza y controla los cambios ocurridos al SW durante el ciclo de vida, para maximizar la productividad y disminuir los defectos. La gestión de configuración del software no es un mantenimiento del software, la GCS es un conjunto de actividades de seguimiento y control que comienzan cuando se inicia el proyecto de desarrollo del SW.

Dos conceptos a tomar en cuenta en éste proceso, es el *elemento de configuración de software* que puede ser un documento, conjunto completo de casos de pruebas, un procedimiento, entre otro. Y *línea base* es una especificación o producto revisado y que se ha decido que en lo adelante servirá como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambios.

Las actividades y productos de este subproceso es expuesta en la tabla 5.3.

Tabla 5.3 Gestión de la Configuración del Servicio - Procesos Gerenciales(a).

Gestión de la Configuración del Servicio			
Actividad	**Tareas**	**Técnica**	**Producto**
Identificación de la Configuración	1. Plan de Gestión de Configuración. a. descripción de actividades de Gestión de Configuración.	Técnicas de GCS	Plan de Gestión de configuración.

Tabla 5.3 Gestión de la Configuración del Servicio - Procesos Gerenciales (b).

Gestión de la Configuración del Servicio			
Actividad	**Tareas**	**Técnica**	**Producto**
	b. procedimientos y cronograma que contempla los responsables de las actividades. 2. Identificación de la Configuración: Establece un esquema de identificación de los elementos de configuración y las versiones a ser controladas por el proyecto.		
Control de la Configuración	1. Identificación y registro de las solicitudes de cambio. 2. Análisis y evaluación de los cambios, donde se aprueba o rechaza la solicitud. 3. Implementación, verificación y distribución del elemento de software modificado.	Técnicas de GCS	Planilla de control de configuración
Contabilidad del estado de configuración	Preparación de registros de gestión y reportes de estado que muestren el estado e historia de los elementos de software controlados, incluyendo líneas base.	Técnicas de GCS	Informe de contabilidad de estado de configuración
Evaluación de la Configuración	Determinación y aseguramiento de los elementos de software sean funcionales (corresponden a los requerimientos) y físicamente completos (diseño y código reflejan una descripción técnica actualizada).	Técnicas de GCS	Informe de la evaluación de la configuración.
Gestión de actualización y distribución	Control formal de la actualización y distribución de los productos de software.	Técnicas de GCS	Informe de Gestión de actualización y distribución de los productos de software.

5.3.4 Verificación y Validación

Este proceso define las actividades a seguir para verificar y validar los productos resultantes con la finalidad de cumplir con los requerimientos y las especificaciones.

Las actividades y productos de este subproceso es visualizada en la tabla 5.4.

Tabla 5.4 Verificación y Validación - Procesos Gerenciales.

Verificación y Validación			
Actividad	**Tareas**	**Técnica**	**Producto**
Planeación de la validación y la verificación	1. Elaboración de planes de validación y verificación para cada etapa. 2. Planeación de los tipos de pruebas.	Modelo Basado en la Verificación (MBV)	Plan de Revisión Técnica Planes de Pruebas
Revisión de los productos	1. Planificar la revisión técnica de los productos. 2. Ejecución de las revisiones. 3. Documentación de los resultados de las revisiones.	Estándares de calidad	Resultados de las revisiones

5.3.5 Gestión del Riesgo

En este proceso se establecen y se aplican estrategias de riesgos sobre el proyecto de SW con la finalidad de identificar eventos potenciales, que pudieran afectar el buen funcionamiento del proyecto.

A través de la tabla 5.5 se muestran las actividades y productos de este subproceso.

Tabla 5.5 Gestión del Riesgo- Procesos Gerenciales

Gestión del Riesgo			
Actividad	**Tareas**	**Técnica**	**Producto**
Identificación de riesgos	Identificación de riesgos y sus costos, esfuerzo y recursos asociados al desarrollo del proyecto.	Software Risk Evaluation SRE services	Lista de riesgos
Análisis de riesgos	1. Determinar la probabilidad de ocurrencia, impacto, tiempo en que puede ocurrir. 2. Determinar técnicas para mitigar el riesgo.	Continuous Risk Management Guidebook	Probabilidades de cada uno de los riesgos y las técnicas de mitigación.
Priorizar los riesgos	Determinar la prioridad en cuanto a la utilización de recursos para mitigarlos.	Software Risk Evaluation Services	Determinación de recursos para mitigación de riesgos
Gestión de riesgos	1. Definir métricas para medir cada riesgo. 2. Definir cuales son los riesgos más peligrosos. 3. Determinar las estrategias de mitigación de riesgos y cual es el más factible. 4. Determinar cuando mitigar estos riesgos.	PMBOK	Planes de mitigación de riesgos
Planeación de la solución de riesgos	1. Tomar acciones correctivas. 2. Determinar si la estrategia de mitigación funcionó o hace falta una nueva o ajustar la existente. 3. Documentar las acciones tomadas y los resultados.	PMBOK	Planes de mitigación de riesgos aplicados

5.3.6 Documentación

Proceso que verifica la generación y consistencia de los diferentes documentos que hay que producir durante el desarrollo de SW. La tabla 5.6 muestra las actividades y productos de éste subproceso.

Tabla 5.6 Documentación - Procesos Gerenciales

Documentación			
Actividad	**Tareas**	**Técnica**	**Producto**
Documentación del SW	Documentación del proceso	Técnicas de Documentación	Procesos documentados
Verificar la consistencia del documento	Verificación de la documentación corresponde con el proceso y revisar si sigue el procedimiento correcto	Técnicas de Documentación	Procesos documentados

5.3.7 Capacitación Técnica

Proceso que comprende la capacitación técnica de grupo de desarrollo del SW, con la finalidad de contar con personal calificado para el desarrollo del proyecto. La tabla 5.7 muestra las actividades y productos de éste subproceso.

Tabla 5.7 Capacitación Técnica (Procesos Gerenciales).

Capacitación Técnica			
Actividad	**Tareas**	**Técnica**	**Producto**
Planificación de la capacitación	Identificación de las necesidades de capacitación. Determinar si la capacitación puede ser hecho por la misma empresa o es necesario contratar otra empresa.	Técnicas de capacitación	Horario de capacitación, selección de personas a entrenar.
Capacitación del grupo de desarrollo	Capacitación del personal.. Mantener un registro de la capacitación recibida por el personal.	Capacitación del personal	Personal capacitado
Producción de las guías de capacitación	1. Elaboración de las guías de capacitación. 2. Distribución de las guías de capacitación por vía electrónica o escrita. 3. Mantenimiento de la documentación.	Técnicas de documentación	Documentación y guías para la capacitación

5.4 Procesos Técnicos

Los procesos técnicos forman un conjunto de fases que permite caracterizar el desarrollo de un SW durante su ciclo de vida, el cual se inicia con la especificación de un SW hasta su publicación en un registro UDDI.

A continuación, se detallan cada una de las fases que conforman los procesos técnicos de desarrollo de un SW. Para cada fase describimos los productos que ella genera, el personal que trabajan durante la fase y su flujo de trabajo.

5.4.1 Fase 1: Especificación del SW

Esta fase tiene como objetivo realizar la especificación de un SW a ser desarrollado, donde la especificación debe representar el SW a desarrollar y los clientes potenciales.

***Productos**:* Documento del diseño de la arquitectura del SW (DDASW), Documento del diseño de flujo de trabajo del SW (DDFTSW), Documento de especificación informal del SW (DEISW), Documento Especificación Formal del SW (DEFSW) y Documentos técnicos de la plataforma (DTP).

Personal: Tipo de personal Nº 2 y Nº 3 (ver capítulo VI – Modelo de Grupo de Desarrollo).

Flujo de trabajo: La secuencia de trabajo de esta fase está ilustrada en la Figura 5.4 que representa la especificación del SW, el detalle de cada subproceso que conforma esté proceso se visualiza mediante las figuras sucesivas; la Figura 5.5 detalla como se define el SW, la Figura 5.6 desglosa el proceso de especificación del SW y la Figura 5.7 representa el proceso de tecnología. La descripción de cada secuencia se muestra mediante la Tabla 5.8.

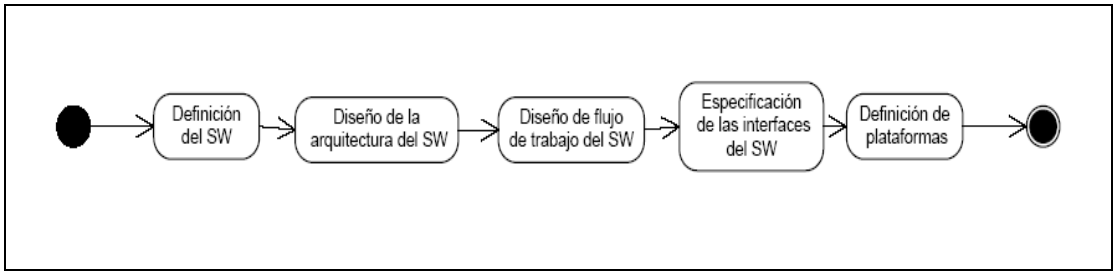

Figura 5.4 Especificación del SW.

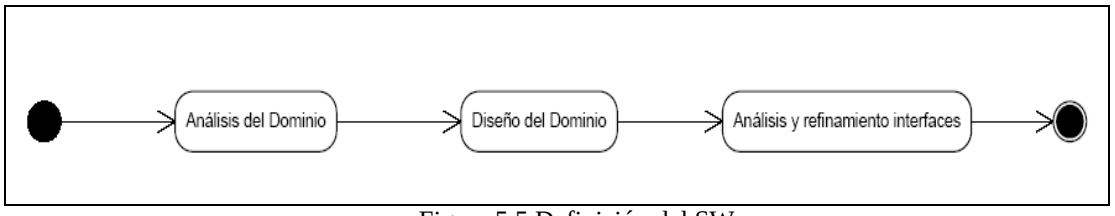

Figura 5.5 Definición del SW.

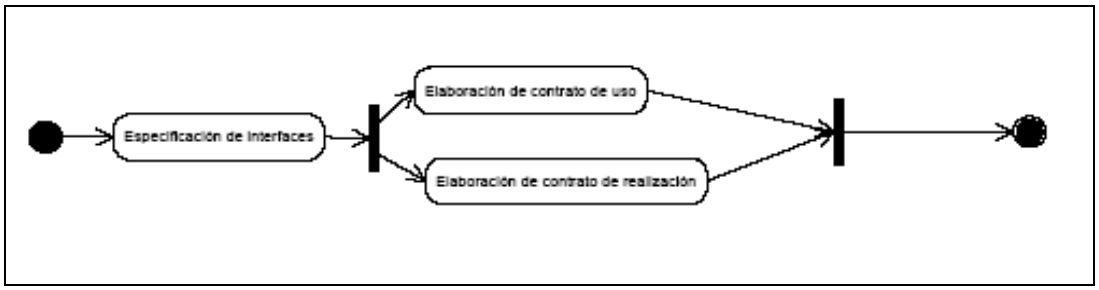

Figura 5.6 Especificación del SW.

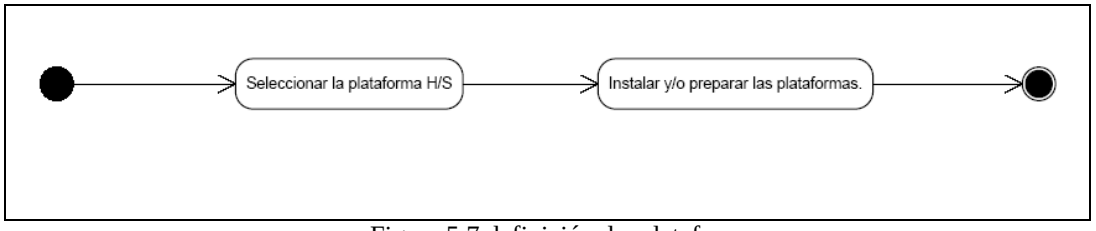

Figura 5.7 definición de plataformas.

Tabla 5.8 Actividades de la Especificación del SW(a).

Actividad	**Tareas**	**Técnica**	**Producto**
Definición del SW	1. Análisis del dominio: consiste en delimitar el dominio y adquisición de conocimiento del dominio.	Características del contexto, mediante: casos de uso, diagramas de colaboración.	Documento de especificación informal del SW (DEISW).

Tabla 5.8 Actividades de la Especificación del SW(b).

Actividad	Tareas	Técnica	Producto
	2. Análisis de la arquitectura del dominios: proceso donde se debe hacer un análisis de los requisitos funcionales, análisis de las conexiones del SW y analizar y clasificar las dependencias del SW con los otros SW. 3. Análisis y refinamiento de las interfaces y operaciones del SW.	Modelo para clasificar las interfaces de los SW propuesto por (Yacoub, Ammar & Mili, 1999).	
Diseño de la arquitectura del SW(Composición del SW)	Diseñar la arquitectura de componentes del SW.	UML component (Cheesman & Daniels, 2001)	Documento del Diseño de la arquitectura del SW (DDASW).
Diseño de flujo de trabajo del SW(Configuración del SW)	1. Identificar otros SW y/o clases de objetos requeridas para implementar el SW. 2. Especificar el flujo de trabajo de los componentes. arquitectónicos del SW. 3. Definir la composición y coreografía de los SW y/o componentes que integran el SW. 4.	UML component (Cheesman & Daniels, 2001)	Documento del Diseño de flujo de trabajo del SW (DDFTSW).
Especificación de las interfaces del SW	1. Especificación de las interfaces: se debe especificar cada operación: entradas, salidas, cambios de estados, invariantes. 2. Describir la interfaz en WSDL. 3. Realizar un modelado de información de la interfaz. 4. Elaboración del contrato de uso. 5. Elaboración del contrato de realización.	UML component (Cheesman & Daniels, 2001)	Documento Especificación Formal del SW (DEFSW)

Tabla 5.8 Actividades de la Especificación del SW(c).

Actividad	Tareas	Técnica	Producto
Definición de Plataformas	1. Seleccionar las plataformas H/S requeridas para desarrollar y operar el SW. 2. Instalar y/o preparar las plataformas.	Analizar la plataforma existente H/S y verificar los elementos de H/S que se requieren.	Documentos técnicos de la plataforma(DTP).

5.4.2 Fase 2: Aprovisionamiento de los componentes del SW

En esta fase se describe el proceso de aprovisionamiento de un SW antes especificado y la definición de los documentos generados en esta fase.

Productos: Catálogos de SW encontrados, Catálogos de SW y la empresa corredoras o desarrolladoras, Documento de resultados de la comparación entre los SW y las técnicas seleccionadas, Catálogo de SW probables, Documentación del SW seleccionado.

Personal: Tipo de personal Nº 4A y Nº 4B (ver capítulo VI – Modelo de Grupo de Desarrollo).

Flujo de trabajo: La secuencia de trabajo de esta fase está ilustrada en las Figuras 5.8, 5.9, 5.10, 5.11, 5.12 y 5.13 y el desglose de cada secuencia es muestra mediante la Tabla 5.9.

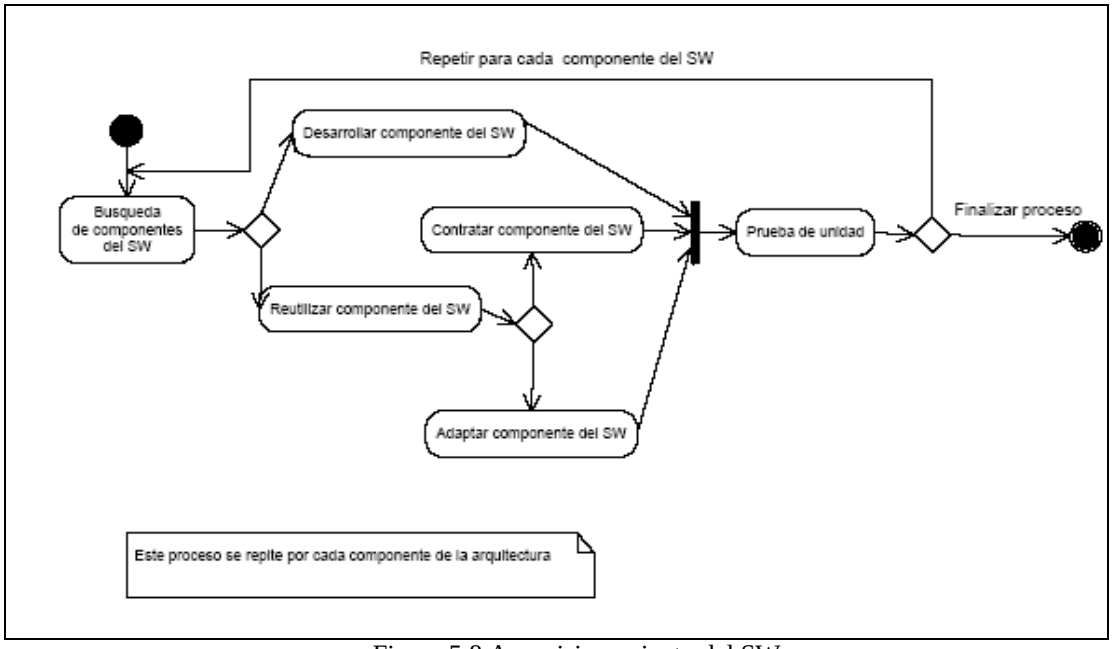

Figura 5.8 Aprovisionamiento del SW.

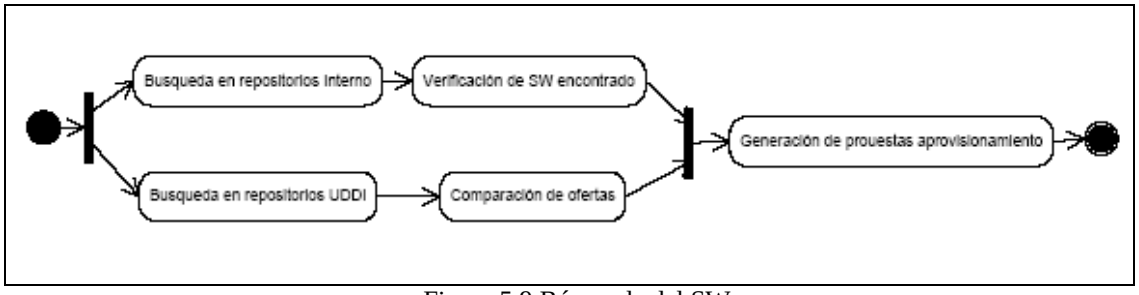

Figura 5.9 Búsqueda del SW.

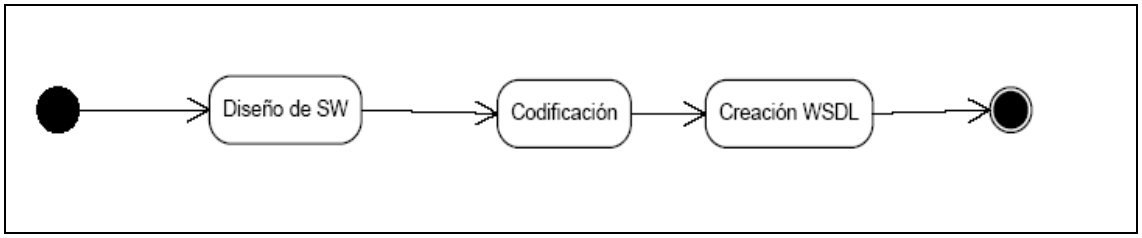

Figura 5.10 Desarrollar componente del SW.

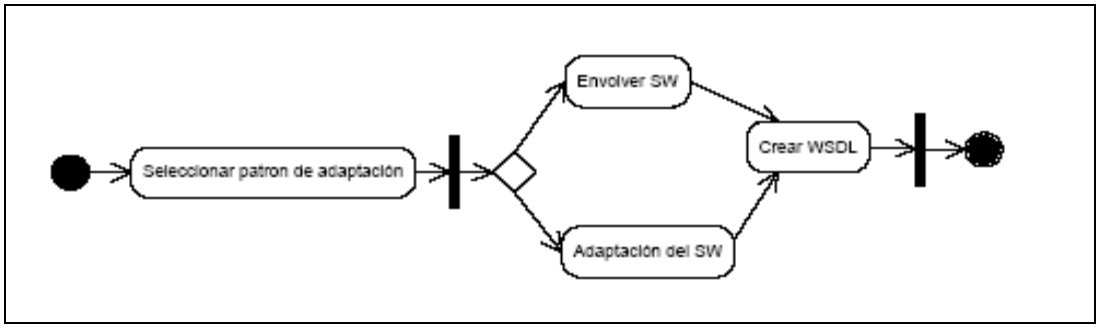

Figura 5.11 Adaptar componente del SW .

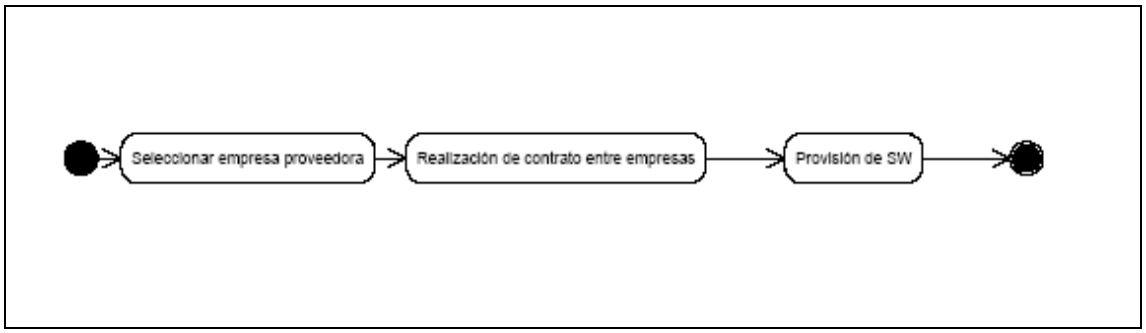

Figura 5.12 Contratar componente del SW.

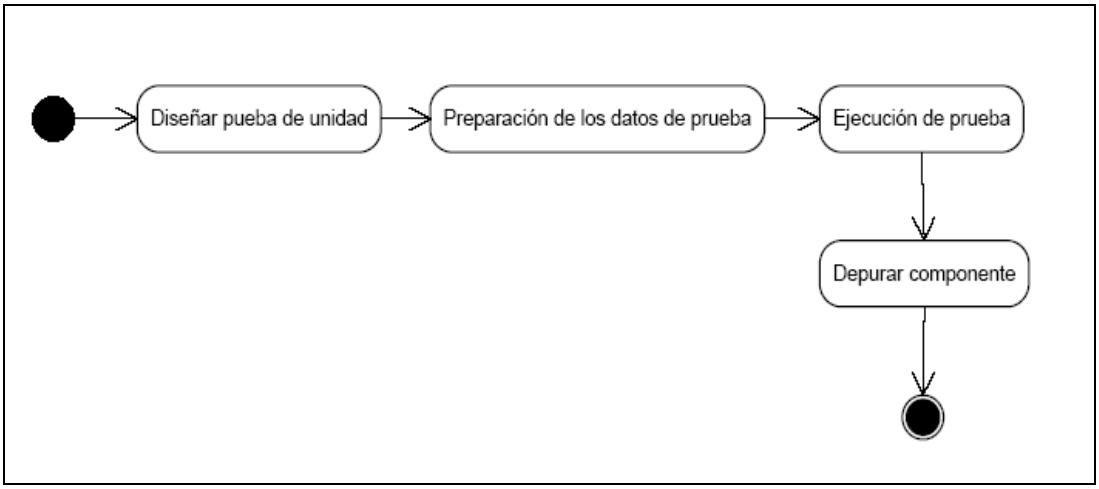

Figura 5.13 Prueba de unidad.

Tabla 5.9 Actividades del Aprovisionamiento del SW(a).

Actividad	**Tareas**	**Técnica**	**Producto**
Buscar de componentes del SW	1. Búsqueda y selección de los componentes que integran la arquitectura del SW en los repositorios externos o internos.	1. Búsqueda en catálogos.	1. Catálogos de SW encontrados.

Tabla 5.9 Actividades del Aprovisionamiento del SW(b).

Actividad	**Tareas**	**Técnica**	**Producto**
	2. Búsqueda y selección de compañías que puedan proveer los componentes arquitectónicos y los costos del mismo.	2. Contactar a las compañías desarrolladoras o corredoras de SW.	2. Catálogos de SW y la empresa corredoras o desarrolladoras.
	3. Verificar la certificación de los componentes arquitectónicos seleccionados y la seriedad de las compañías proveedoras y dependencias del SW.	3. Técnicas de comparación (Análisis GAP) Tablas de comparación con los factores más relevantes del proyecto.	3. Documento de resultados de la comparación entre los SW y las técnicas seleccionadas.
	4. Generar las diferentes propuestas para el aprovisionamiento del SW	4. Análisis de Riesgos.	4. Catálogo de SW probables.
Selección de la forma de aprovisionamiento del SW.	1. Elegir patrones de comparación entre las opciones generadas en el paso anterior: Deben cumplir con los requisitos.	1. Análisis GAP (permite encontrar la diferencia entre los requisitos y los SW existentes) y patrones de comparación propuestos por (PerOlof, 1997).	1. Patrones de comparación de SW.
	2. Seleccionar la técnica de aprovisionamiento.	2. Estudio de resultados análisis GAP.	2. Resultados de la comparación de los SW.

Tabla 5.9 Actividades del Aprovisionamiento del SW(c).

Actividad	**Tareas**	**Técnica**	**Producto**
Desarrollar / Adaptación de componente del SW	1. Escoger la técnica de aprovisionamiento del SW.	1. Tomar decisiones de acuerdo a los resultados de la comparación.	1. Decisión acerca de la forma de aprovisionamiento.
	2. Adaptar. Escoger el patrón de adaptación, adaptar SW, crear el archivo WSDL.	2. Patrones de adaptación Técnicas de adaptación (Wrap, entre otras).	2. SW Adaptado.
	3. Desarrollar. Diseñar SW, codificar SW, crear archivo WSDL.	3. Herramientas para crear WSDL 4. Técnicas de programación OO. 5. Herramientas para crear WSDL	3. SW Desarrollado.
	4. Contratar. Selección de la empresa que realice el aprovisionamiento, realizar el contrato entre empresas y aprovisionamiento del SW.	6. Estudio de las formas de contratación de las empresas corredoras o desarrolladoras.	4. SW provisto por terceros.
Prueba de unidad	1. Diseñar prueba de unidad. 2. Preparación de los datos de prueba. 3. Ejecución de prueba 4. Depurar componente	Realizar prueba de unidad	Componente del SW probado.
Documentación del aprovisionamiento	Documentación del SW seleccionado, los pasos realizados para tener el SW esperado.	Documentación en lenguaje forma UML.	Documentación del SW seleccionado

5.4.3 Fase 3: Ensamblaje del SW

Ésta fase tiene como objetivo ensamblar el SW desarrollado con los componentes que lo integran.

***Productos**:* SW Ensamblado.

*Personal***:** Tipo de personal Nº 4A y Nº 4B (ver capítulo VI – Modelo de Grupo de Desarrollo).

Flujo de trabajo: La secuencia de trabajo de esta fase está ilustrada en las Figuras 5.14 y 5.15 y el desglose de cada secuencia es muestra mediante la Tabla 5.10.

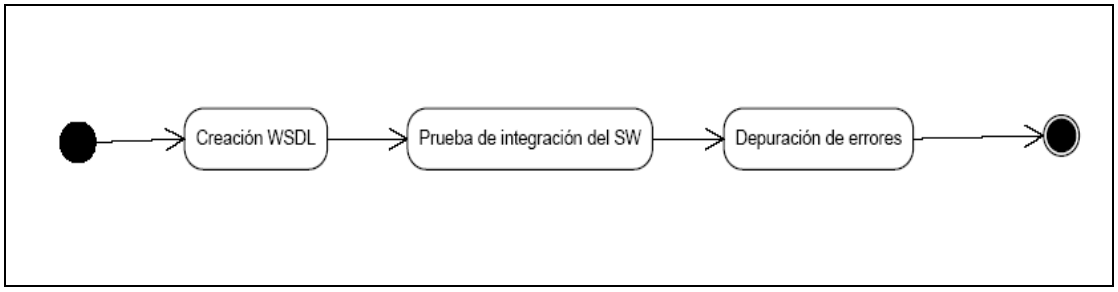

Figura 5.14 Ensamblaje del SW.

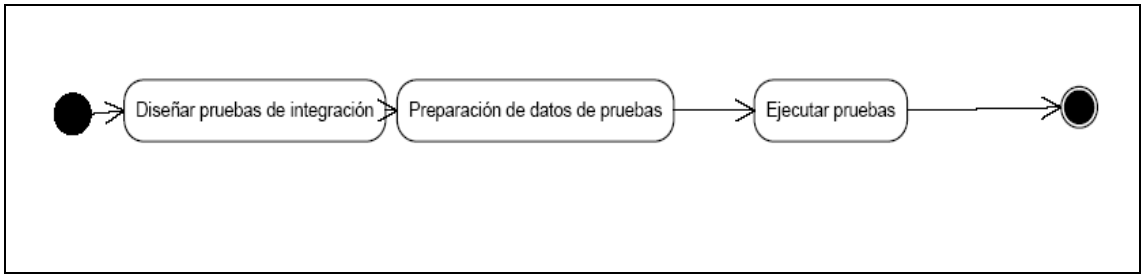

Figura 5.15 Prueba de integración del SW.

Tabla 5.10 Actividades Ensamblaje del SW.

Actividad	Tareas	Técnica	Producto
Creación de WSDL.	Crear el archivo WSDL.	Herramientas para crear WSDL.	Archivo WSDL.
Pruebas de integración del SW.	1. Diseñar las pruebas de integración del SW. 2. Preparación de los datos para realizar las pruebas. 3. Ejecutar las pruebas.		Pruebas de integración realizadas.
Depuración de errores.	Realizar la depuración de errores que resultaron del paso anterior.		SW ensamblado.

5.4.4 Fase 4: Pruebas del SW

Las pruebas tienen como objetivo fundamental conseguir las diferencias entre la manera esperada del SW y la forma observada en el sistema.

Existen diferentes pruebas de software, entre ellas tenemos: pruebas unitarias donde se encuentra diferencias entre el modelo de diseño de objetos y sus componentes correspondientes. Las pruebas estructurales encuentran diferencias entre el modelo del diseño del sistema y un subconjunto de subsistema integrados. Las pruebas funcionales encuentran diferencias entre el modelo de caso de uso y el sistema y las pruebas de desempeño encuentran diferencias entre los requisitos no funcionales y el desempeño real del sistema. (Brugge y Dutoit, 2000).

El objetivo de la fase es proporcionar un conjunto de actividades y técnicas que den como resultado el plan, diseño y ejecución de las pruebas, por parte del desarrollador del SW con miras a demostrar que no se presentan errores o identificar errores, producto de las pruebas realizadas.

Dentro de las pruebas recomendadas a realizar están:

Pruebas funcionales: encargada de comprobar que el SW cumple con las funciones establecida en los contratos de uso y de realización.

Pruebas no funcionales: realiza la comparación del SW con los requisitos no funcionales, estos incluyen seguridad, velocidad, confiabilidad, entre otros.

Pruebas de aceptación: consiste en revisión de los requisitos contra el contrato de uso; esta prueba debe ser realizada por el cliente o usuario.

Producto: Documento de pruebas a realizar, Documento de pruebas funcionales, Documento de pruebas de comportamiento, Documento de pruebas de aceptación, Documento del análisis de los resultados, Documentos de las pruebas y resultados SW probado

Personal: Tipo de personal Nº 5 (ver capítulo VI – Modelo de Grupo de Desarrollo).

Flujo de trabajo: La secuencia de trabajo de esta fase esta ilustrada en las Figuras 5.16, 5.17, 5.18, 5.19 y 5.20, y el desglose de cada secuencia es muestra mediante la Tabla 5.11.

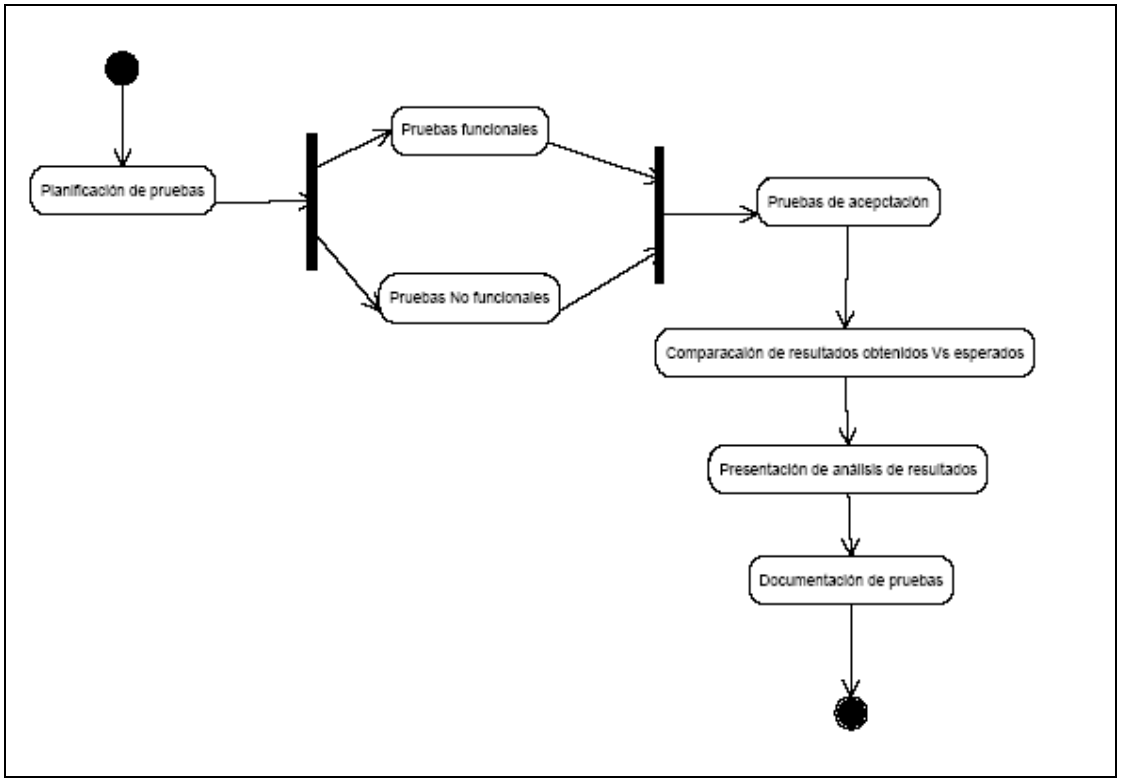

Figura 5.16 Pruebas del SW.

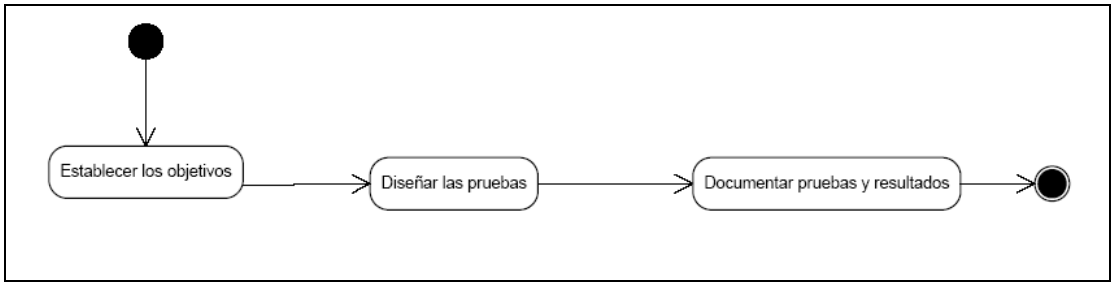

Figura 5.17 Planificación de pruebas del SW.

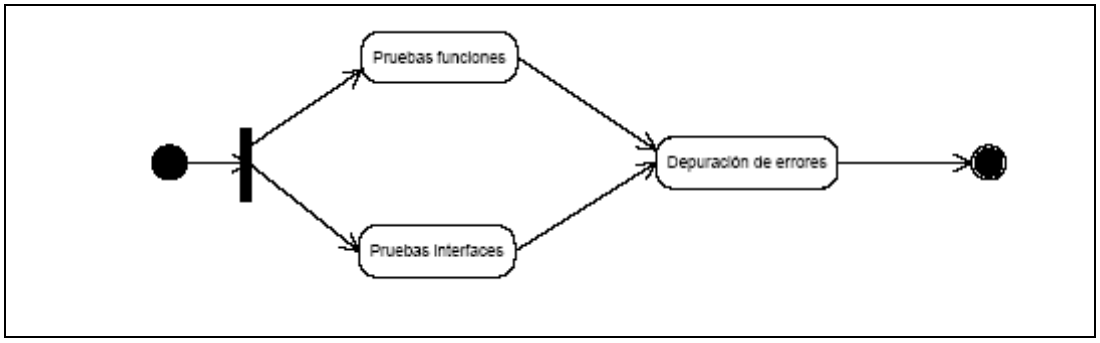

Figura 5.18 Pruebas funcionales del SW.

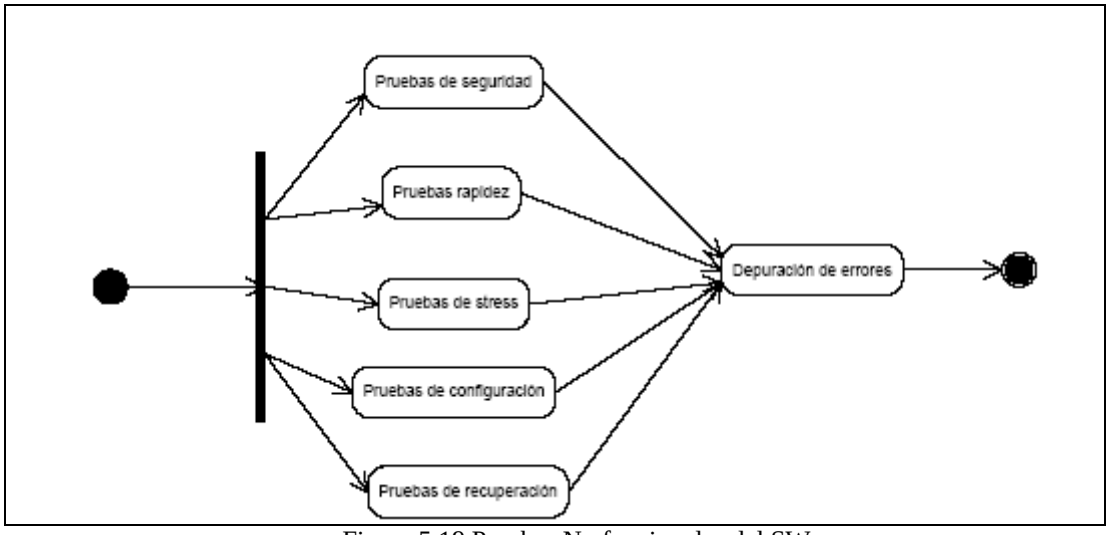

Figura 5.19 Pruebas No funcionales del SW.

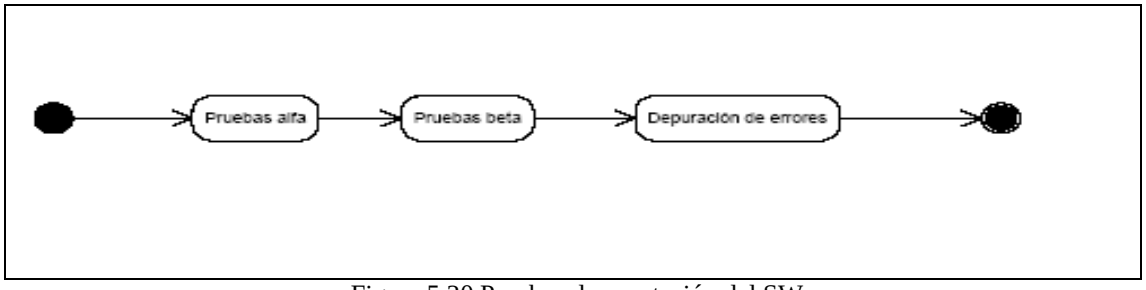

Figura 5.20 Pruebas de aceptación del SW.

Tabla 5.11 Actividades Pruebas del SW(a).

Actividad	**Tareas**	**Técnica**	**Producto**
Planificación de las pruebas	Establecer los objetivos. Diseñar las pruebas. Documentar cada uno de los casos de pruebas y los resultados esperados.	Planificación de las pruebas de acuerdo al contrato de uso y al contrato de realización	Documento de pruebas a realizar.

Tabla 5.11 Actividades Pruebas del SW(b).

Actividad	Tareas	Técnica	Producto
Realizar pruebas funcionales	1. Pruebas de las funciones. 2. Pruebas de las interfaces.	Rational Test. Pruebas. automáticas Pruebas manuales.	Documento de pruebas funcionales.
Realizar pruebas no funcionales)	1. Pruebas de seguridad. 2. Pruebas de rapidez. 3. Pruebas de confiabilidad. 4. Pruebas de stress. 5. Pruebas de volumen. 6. Pruebas de configuración. 7. Pruebas de recuperación.	Pruebas automáticas. Pruebas manuales.	Documento de pruebas de comportamiento.
Realizar pruebas de aceptación	1. Pruebas alfa. 2. Pruebas beta.	Pruebas en ambiente de desarrollo.	Documento de pruebas de aceptación
Evaluación de las pruebas de resultados	1. Comparar los resultados obtenidos con los esperados. 2. Presentar el análisis de los resultados.	Técnicas estadísticas. Comparaciones.	Documento del análisis de los resultados.
Documentación de las pruebas	Documentar las pruebas	Documentación interna.	Documentos de las pruebas y resultados SW probado

5.4.5 Fase 5: Despliegue del SW

La fase de despliegue incluye la publicación de la interfaz y la definición de la implementación del SW en un registro del servicio UDDI interno y despliegue de los ejecutables del SW dentro de un ambiente de ejecución (Servidor de Aplicaciones Web). AL finalizar esta fase el SW esta completamente operacional pero puede ser utilizado solo por usuarios internos.

En está fase del desarrollo de un SW es recomendable realizar un diagrama de despliegue, con la finalidad de mostrar la infraestructura completa donde reside y se despliega los SW. A través de la Figura 5.21, se muestra un ejemplo de la arquitectura de SW que ilustra el objetivo de la fase.

Producto: Documento de la fase de despliegue.

Personal: Tipo de personal Nº 6 (ver capítulo VI – Modelo de Grupo de Desarrollo).

*Flujo de trabajo***:** La secuencia de trabajo de esta fase es visualizada en la Figura 5.22 y el detalle de cada actividad es muestra mediante la Tabla 5.12.

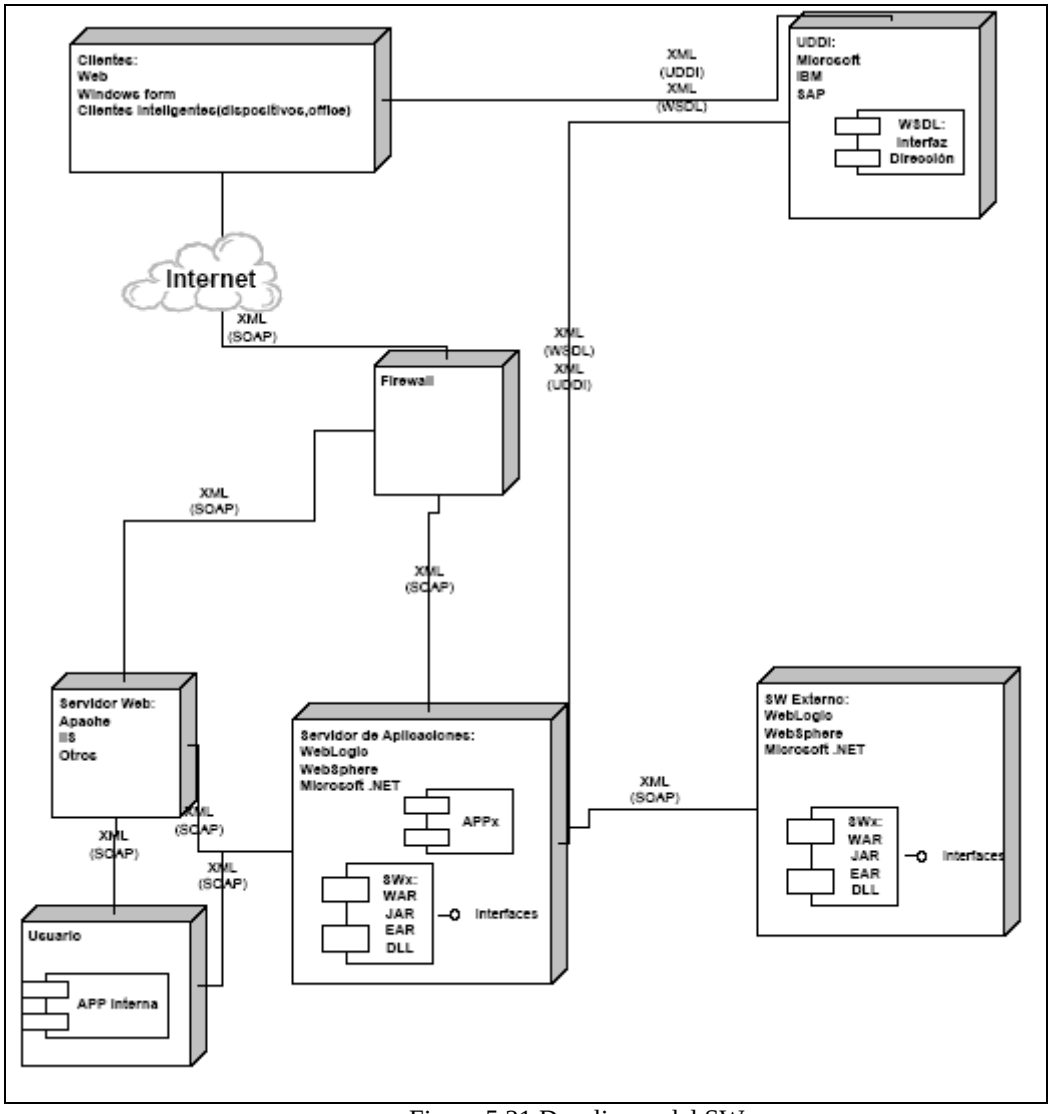

Figura 5.21 Despliegue del SW.

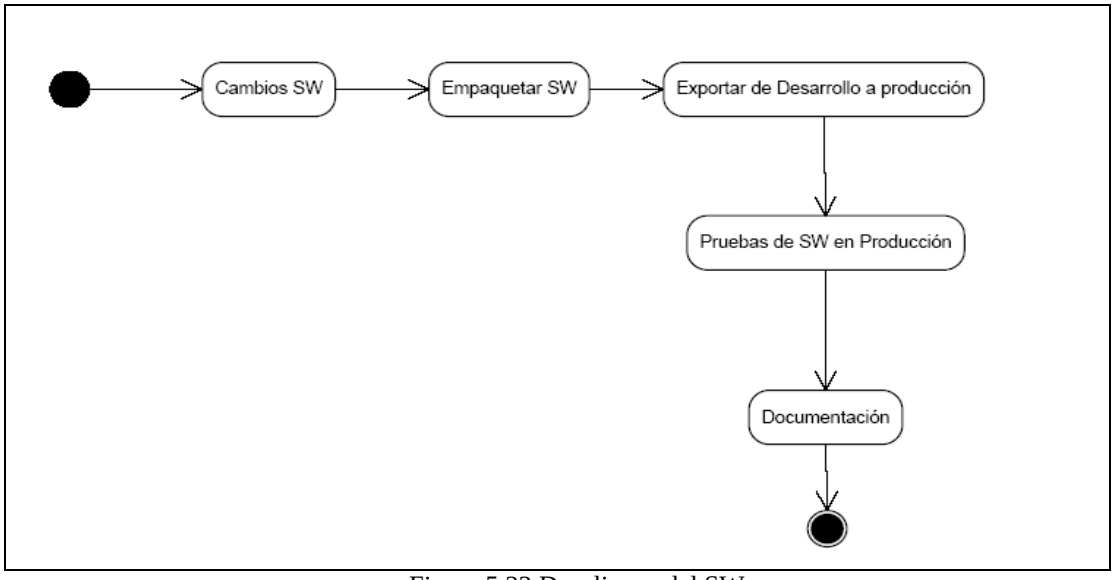

Figura 5.22 Despliegue del SW.

Tabla 5.12 Actividades de Despliegue del SW.

Actividad	**Tareas**	**Técnica**	**Producto**
Cambiar SW	Cambio de Puerto y URL	Utilizar software seleccionado	SW modificado.
Empaquetar SW	De acuerdo a la plataforma, empaquetar, esto significa que: 1. En J2EE, esto significa: JAR, WAR y EAR . 2. En .NET, esto significa CAB, .NET Assemblies, entre otros.	Utilizar software seleccionado	SW Empaquetado.
Exportar SW	Trasladar archivos necesarios de ambiente de desarrollo a producción.	Utilizar software seleccionado	SW Exportado.
Pruebas de despliegue	Pruebas de funcionamiento del SW en el ambiente de producción	Utilizar software seleccionado	SW probado
Documentación	Documentar las actividades realizadas.	Documentación interna	Documento de la fase de despliegue.

5.4.6 Fase 6: Publicación del SW

La fase de publicación consiste en registrar el SW en un repositorio externo conjuntamente con la implementación, esto con la finalidad que el SW este operacional y accesible en la red desde el Proveedor de Servicio. Ahora el solicitante del servicio puede realizar la búsqueda y efectuar operaciones.

Producto: Documento de la fase de Registro.

Personal: Tipo de personal Nº 6 (ver capítulo VI – Modelo de Grupo de Desarrollo).

Flujo de trabajo: La secuencia de trabajo de esta fase esta ilustrada en las Figuras 5.23 y 5.24, y el desglose de cada secuencia es muestra mediante la Tabla 5.13.

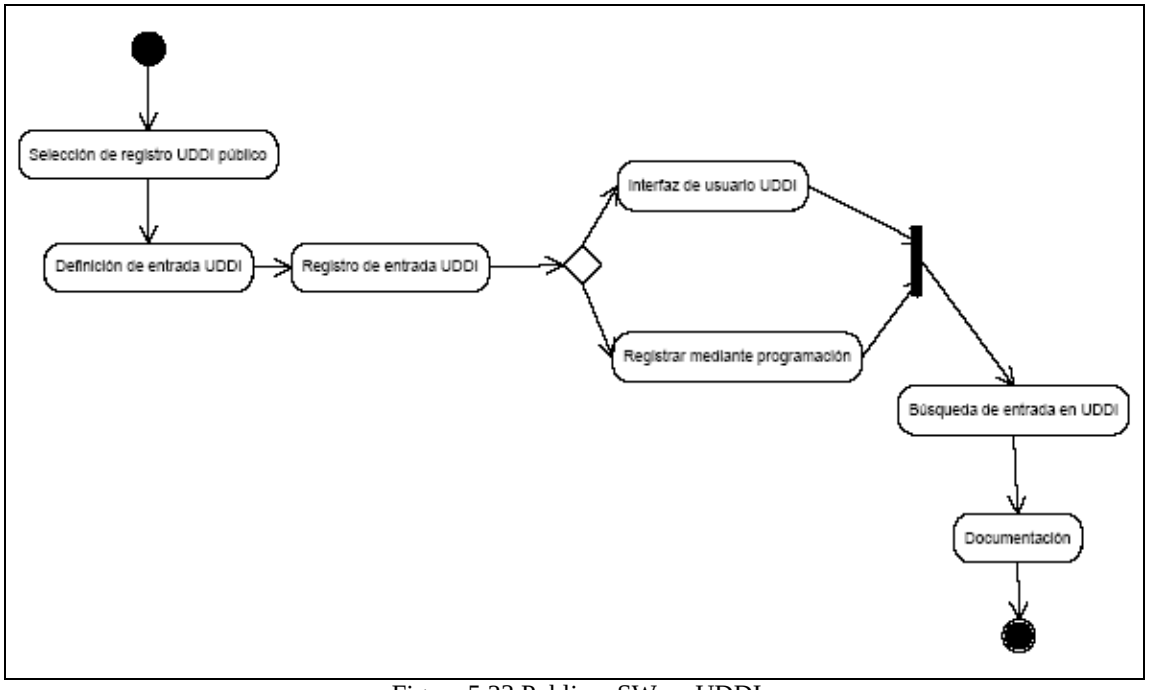

Figura 5.23 Publicar SW en UDDI.

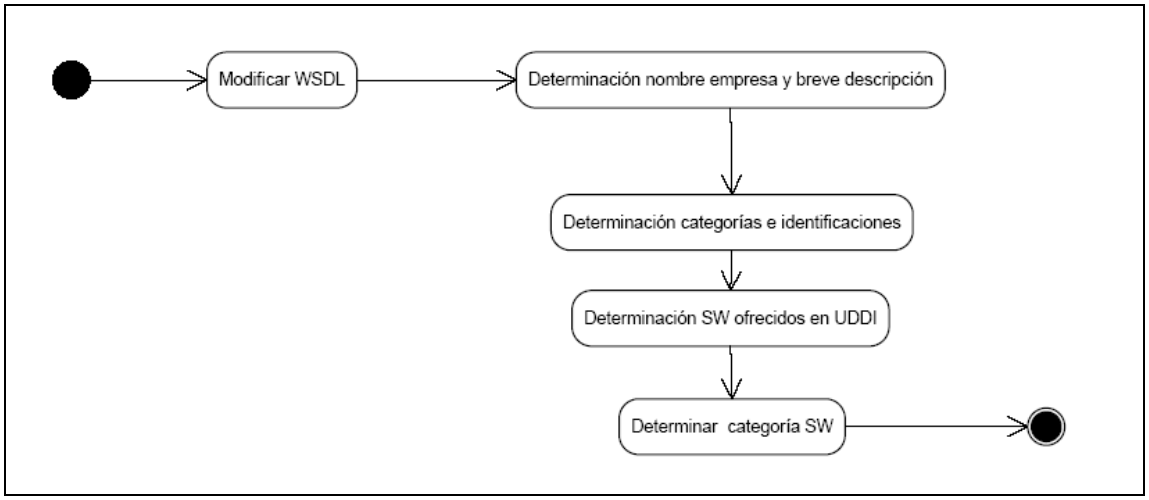

Figura 5.24 Definición de entrada UDDI.

Tabla 5.13 Actividades publicación del SW(a).

Actividad	**Tareas**	**Técnica**	**Producto**
Selección de registro UDDI	Escoger en que UDDI público(IBM, Microsoft, entre otros) se va a registrar el SW.	Escogencia de registro UDDI	
Definición de entrada UDDI	1. Determinar los tModels (archivos WSDL). 2. Determinar el nombre de la empresa y una breve descripción de la misma en varios idiomas 3. Determinar las categorías e identificaciones adecuadas para la empresa. 4. Determinar SW que la empresa ofrece a través de UDDI. 5. Determinar las categorías adecuadas para los SW.	Herramientas para crear WSDL	Archivo WSDL, documentación de empresa
Registro de entrada	1. Utilizar interfaz de usuario UDDI. 2. Publicar mediante programación.	1. Usar las interfaz del registro UDDI seleccionado. 2. Utilizar la interfaz desarrollada.	SW Registrado.

Tabla 5.13 Actividades publicación del SW(b).

Actividad	**Tareas**	**Técnica**	**Producto**
Búsqueda de entrada en UDDI	3. utilizar la interfaz de usuario Web UDDI, busque la empresa por su nombre y categorizaciones para verla entre los conjuntos de resultados devueltos. 4. Transcurridas las 24 horas, la entrada se replicará otros nodos UDDI, buscar con su interfaz de usuario	Interfaz creada.	SW localizado.
Documentación	Documentar las actividades realizadas.	Documentación interna	Documento de la fase de Registro.

CAPÍTULO VI

MODELO DE GRUPO DEL MÉTODO DESWEB

Este capítulo trata lo referente al modelo de grupo del método DESWeb. El modelado se describe a través de la identificación de los roles que deben ser jugados por los diferentes miembros del grupo y sus tareas en cada fase del proceso de desarrollo de un SW.

La estructura del capítulo comprende dos secciones: sección 6.1 describe el grupo de desarrollo y la sección 6.2 se muestra la estructura del grupo de desarrollo del método.

6.1 Grupo de Desarrollo

El recurso humano es factor importante al emprender el desarrollo de un proyecto de software, donde se debe determinan el personal, responsabilidad y funciones que debe realizar durante el desarrollo del proyecto. De acuerdo a Zimmermann y Muller (2004), los diversos roles de trabajo implicados en el desarrollo de proyectos de Software de SW se encuentran:

1. *Encargado del proyecto*. Responsable de la gerencia y dirección del equipo de proyecto. Define y sigue la estructura del plan de trabajo del proyecto. Igualmente trabaja la estructura de algunas eventualidades no contempladas en el proyecto original.

2. *Analista del negocio*. Encargado de realizar el levantamiento de los requisitos funcionales y no funcionales de los usuarios del negocio, así como las restricciones y proporcionar conocimientos del dominio al equipo de trabajo. Debe entender la lógica del negocio y tener experiencia en trabajo similares.

3. *Arquitecto SOA*. Responsable del servicio final del solicitante y el diseño del proveedor. Investiga y indica los requisitos no funcionales del servicio.

4. *Analista de despliegue*. Encargado de exporta artefactos del ambiente de desarrollo y los instala en el ambiente de ejecución. Igualmente genera los matrices y plantillas de WSDL para el ambiente de ejecución y los instala conjuntamente con implementaciones de servicio.

5. *Facilitador de transferencia de conocimiento*. Es responsable de proporcionar el conocimiento que le falta a los desarrolladores sobre un tema específico. Este tema puede estar relacionado con el método de desarrollo, el proceso, la tecnología de implementación o el ambiente de desarrollo.

6. *Desarrollador de servicio*. Responsable del desarrollo, familiarizado con conceptos de servicios Web y XML. Desarrolla la interfaz del servicio e implementación (lado del proveedor) y código de invocación del servicio (lado del solicitante).

7. *Probador*. Encargado de realizar las pruebas según lo especificado en el diseño y de acuerdo a los estándares por etapas, tales como: integración, carga, y prueba de aceptación. También define los casos de prueba para la interoperabilidad y la conformidad de los Servicios Web.

8. *Probador de la Interoperabilidad*. Responsable de verifica que lo desarrollado por el solicitante y las implementaciones del proveedor aseguren la interoperatividad y la conformidad de la interoperabilidad de los Servicios Web (WS-I).

9. *Administrador UDDI*. Responsable de define cómo el modelo de datos UDDI genérico es modificado para requisitos particulares y generales. Un rol opcional.

10. *Administrador del sistema y base de datos*. Es responsable de realiza la instalación y trabajos de mantenimiento implementados sobre la plataforma - hardware, sistema operativo, base de datos y middleware.

La Figura 6.1 visualiza un Diagrama de jerarquía donde el rol principal recae sobre el Líder del proyecto.

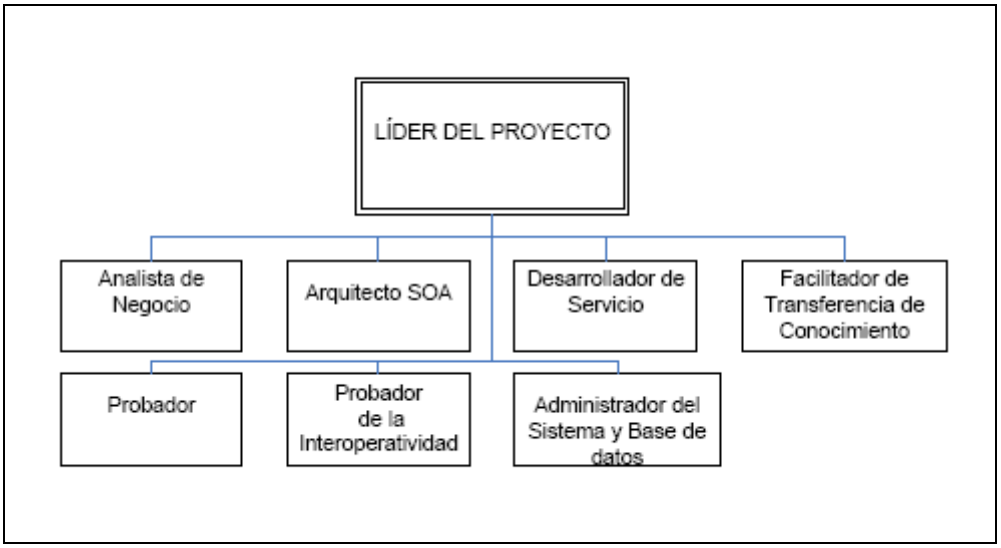

Figura 6.1 Diagrama de jerarquía del Grupo de desarrollo.

6.2 Estructura del Grupo de Desarrollo

Dentro del grupo de desarrollo se encuentran involucrados varios roles, cada rol involucra diversas etapas del proyecto, lo que implica que una persona puede realizar varios roles o funciones dependiendo de las habilidades y conocimientos de la persona y de la disponibilidad de recurso financiero de la organización. En la tabla 6.1 es plasmada una propuesta para un grupo de trabajo para desarrollo de proyectos de pequeña y medina empresa.

Tabla 6.1 Roles y responsabilidad del Grupo de Desarrollo.

Tipo de Personal integrante del Grupo de desarrollo	**Rol asignados**	**Colabora con**	**Tareas Asignadas**
1	Líder del proyecto.	(todos los miembros del equipo).	Planificación del proyecto. Control del proyecto.
2	Analista del negocio	3	Análisis del dominio del problema
3	Arquitecto de SOA. Analista de despliegue. Facilitador de transferencia de conocimiento de SW.	(todos los miembros del equipo).	Arquitectura del software y sistema. Modela el WSDL para las invocaciones del servicio.
4ª	Desarrollador y probador de SW (la unidad).	3, 5, 6.	Localiza de riesgos y administración de políticas de requerimientos de servicio (clientes).
4B	Desarrollador y probador de SW (la unidad).	3, 5, 6.	Codifica y implementa la solicitud del proveedor mediante SOAP.
5	Probador. Probador de la interoperabilidad.	3, 6	Integrar y conectar los componentes desarrollados por 4a y 4b.
6	Encargado de mantenimiento de despliegue. Administrador de sistema. Administrador de Base de datos. Administrador de UDDI.	3, 4x, 5.	Administración de la plataforma donde se desarrolla y ejecutan los SW.

CAPÍTULO VII

VALIDACIÓN DEL MÉTODO DESWEB

Capítulo que tiene por finalidad la evaluación del Método DESWeb en un caso práctico. El caso de estudio es un SW Calculadora, éste servicio expondrá cuatro métodos: Sumar, restar, multiplicar y dividir.

El capítulo contiene tres (3) secciones, la sección 7.1 se comenta en forma general la evaluación del método DESWeb, la sección 7.2 trata de los procesos gerenciales del método y la sección 7.3 se relaciona con los procesos técnicos del método.

7.1 Evaluación del Método DESWeb

.

El empleo del Método DESWeb se inicia con la ejecución de los procesos gerenciales, que son el pilar fundamental del método. Procesos que están relacionados con la gerencia de los Recursos Humano y la Gerencia de la calidad.

Definido y concretado los fines del proyecto, se continúa con los procesos técnicos, donde se hace énfasis en la fase del aprovisionamiento de componente del SW.

7.2 Procesos Gerenciales

Los procesos gerenciales son llevados a cabo por el encargado del proyecto. En este caso práctico se desarrollarán los procesos que se adecuen a las necesidades del caso.

7.2.1 Gestión del proyecto

7.2.1.1 Iniciación del proyecto: El proyecto tiene como objetivo crear un SW Calculadora que contenga las operaciones elementales matemáticas: Suma, resta, multiplicación y división.

Las entradas aceptadas son dos números naturales y generará un número como resultado de la operación realizada.

7.2.1.2 Los requisitos generales del proyecto

1. Personal: Encargado del proyecto y una persona que realizará las funciones del tipo de personal 4,5,6 (de acuerdo a la tabla 6.1 del CAPÍTULO VI del presente trabajo).
2. La plataforma donde se desarrollará y ejecutará el SW.

***7.2.1.3 Actividades relevantes*:** Las tareas más relevantes están enmarcadas en la fase de aprovisionamiento de componente del SW.

7.2.2 Gestión de la Calidad del servicio

Para asegurar la calidad del SW se realizarán y documentará las pruebas necesarias de acuerdo al contrato de uso y el contrato de realización.

7.2.3 Gestión de la Configuración del servicio

En el caso de estudio, se aseguró que se cumplan las características de portabilidad y reusabilidad, que va a permitir que sea adaptado a las diferentes formas de salida y plataformas.

7.2.4 Verificación y Validación

El proceso que se lleva a cabo al final de cada fase. De acuerdo a los resultados se continúa a la próxima fase o se retorna a la fase anterior. Esta será desglosada en cada una de las fases de desarrollo.

7.3. Procesos Técnicos

7.3.1 Fase 1. Especificación del SW

7.3.1.1 Definición del SW

El dominio del SW esta enmarcado en la matemática, pero más exclusivamente dentro de la *Aritmética* operaciones básicas: Suma, resta, multiplicación y división.

El SW utilizará valores enteros naturales para resolver la operación ingresada. La comunicación con el SW es mediante una interfaz que reciba los datos de entrada, generando un valor numérico como resultado de la operación. Ver Figura 7.1

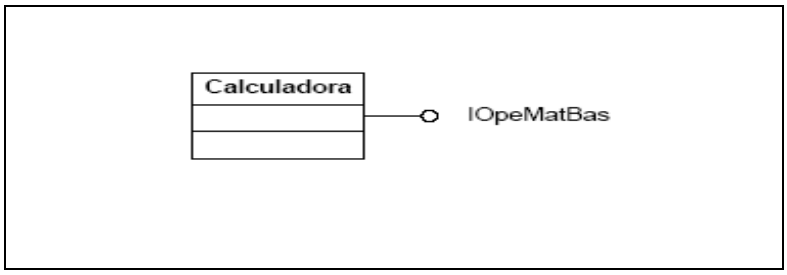

Figura 7.1 Definición general de SW OperMatBas

7.3.1.1.1 Requisitos funcionales

1. El SW recibe dos números enteros naturales como parámetros de entrada.
2. Si la longitud de los números supera los 5 dígitos, el SW retorna un mensaje de error.
3. El SW retorna el resultado, a través de un parámetro de salida.

7.3.1.1.2 Conexiones del SW: El SW operaciones matemáticas puede ser requerido por una aplicación que necesite realizar operaciones básicas de matemáticas, y debe ser invocado a través de la interfaz definida.

7.3.1.1.3 Dependencias del SW: El SW operaciones matemáticas realiza sus funciones sin depender de ningún otro SW.

7.3.1.2 Especificación de SW

a) *Interfaces soportadas*: La interfaz presente en el SW es IOperMatBas
b) *Interacción con otros SW*: Se comunica mediante la única interfaz.
c) Método del SW: Clase Calculadora, contiene los métodos sumar, restar, multiplicar y dividir

d) A través de las figuras Figura 7.2 y 7.3 se visualizan la especificación del SW SWOperMatBas y la definición de la interfaz IOperMatBas respectivamente.

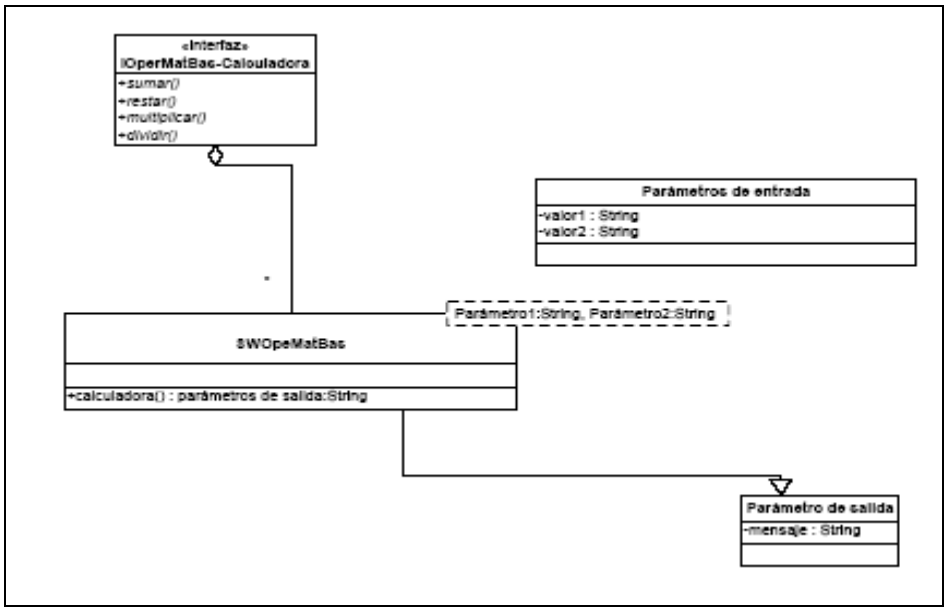

Figura 7.2 Especificación del SW SWOperMatBas

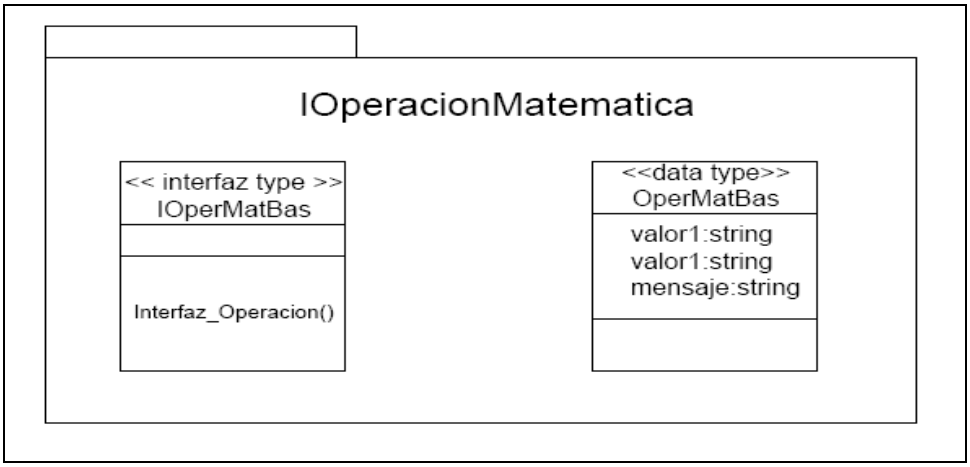

Figura 7.3 Definición de la Interfaz IOperMatBas

7.3.1.2.1 Signaturas: Calculadora()

a) *Excepciones: D*ivisión por cero (0) y operando vacíos.

b) Signaturas de cada una de las operaciones:

Calculadora.sumar(valor1 string, valor2 string)
Calculadora.restar(valor1 string, valor2 string)
Calculadora.multiplicar(valor1 string, valor2 string)

Calculadora.dividir(valor1 string, valor2 string)

b.1) Operaciones:

Precondiciones: permite número naturales enteros

***Post-condiciones*:** La salida es una cadena contenida de un número cuando la operación matemática es exitosa, en caso contrario es una cadena que contiene un mensaje de error.

7.3.1.2.2 Restricciones: El número de valores de entrada son dos(2).

7.3.1.2.3 Comportamiento: Se espera que SW sea invocado por una aplicación y devuelva el resultado de la operación.

7.3.1.2.4 Invariantes: El SW espera ser invocado con los valores de entrada, realiza la operación correspondiente y devolver el resultado de la operación.

7.3.1.2.5 Describir la interfaz en WSDL

La descripción WSDL indica que se soportan cuatro operaciones: Sumar, Restar, multiplicar y dividir. La primera permite la sumatoria de dos números, la segunda realiza la resta de dos números, la tercera hace la multiplicación de dos números y la cuarta realiza la división de dos números.

Las cuatro operaciones tienen dos parámetros de entrada tipo entero, un parámetro de salida tipo string. La salida tomará el resultado de la operación o determinado mensaje de error.

7.3.1.2.6 Contrato de uso

Contrato de Uso: SWOperMatBas

Interfaz Soportada: IOperMatBas

Signatura: Calculadora()

Excepciones: Cuando se divide por cero(0), datos no vacíos.

Signaturas de cada una de las operaciones:

Calculadora.sumar(valor1 string, valor2 string)
Calculadora.restar(valor1 string, valor2 string)

Calculadora.multiplicar(valor1 string, valor2 string)
Calculadora.dividir(valor1 string, valor2 string)

Precondiciones-operaciones: dos valores de entrada.

Post-condiciones-operaciones**:** La salida es una cadena contenida de un número cuando la operación matemática es exitosa, en caso contrario es una cadena que contiene un mensaje de error.

Restricciones: El número de valores de entrada son dos(2).

Comportamiento: Se espera que SW sea invocado por una aplicación y devuelva el resultado de la operación.

Invariantes: El SW espera ser invocado con los valores de entrada, realiza la operación correspondiente y devuelve resultado de la operación.

7.3.1.2.7 Contrato de realización

El SW SWOperMatBas, contiene una especificación del SW y una interfaz.

Los métodos presentes en el SW: Suma, restar, multiplicar y dividir, donde la precondición es que los datos no sean vacíos y en el caso de la división no es permitido que el segundo valor sea cero.

Métodos:

Calculadora.sumar(valor1 string, valor2 string)
Calculadora.restar(valor1 string, valor2 string)
Calculadora.multiplicar(valor1 string, valor2 string)
Calculadora.dividir(valor1 string, valor2 string)

Precondición: Datos de entrada no vacíos y no permite la división por cero.

Postcondición: retorna el valor resultante, en el caso de ocurrir un error devolver un mensaje de error.

7.3.1.3 Definición de plataforma

El SW se desarrollará bajo la siguiente plataforma:

Software: Visual Studio .NET 2003 con Windows 2000 Server con el lenguaje de programación Visual Studio .NET 2003.

Hardware: El servidor en un PIV 3.0 GHz con 1 GB MB de memoria y a nivel del equipo que usa el desarrollador es un PIV 2.8 GHz con 512 MB de memoria.

7.3.2 Fase 2: Aprovisionamiento de componente del SW

7.3.2.1 Búsqueda del SW: Se hizo la búsqueda en repositorio externo y la web, se consiguieron varios SW que cumple con la mayoría de las especificaciones, tal son los casos de:

1. Simple Calculator Web Service:
http://www.xmlwebservices.cc/index_Samples.htm

2. Servicio Matemático simple:
http://es.gotdotnet.com/quickstart/aspplus/doc/writingservices.aspx

Sin embargo se le hizo unas adaptaciones dado que no cumple que ciertas especificaciones, de mostrar mensaje de error cuando se realiza una división por cero o valores vacíos.

7.3.2.2 Adaptación del SW:

Para cada operación matemática: si valor1 = 0 o valor2 = 0 entonces error ” Ningún operando debe ser vacío...”

División: dividir = valor1 / valor2
Si dividir = "Infinito" entonces error "Error división por cero"

7.3.2.3 Crear WSDL:

Se utilizo el comando disco.exe de Visual Studio .Net para generar el WSDL

7.3.2.4 Pruebas de Unidad:

En este caso practico no se hicieron textualmente las prueba que deben realizarse de acuerdo al método dado la sencillez del mismo. Sin embargo se hicieron la prueba de ingresar valores para verificar su operatividad.

7.3.3 Fase 4: Pruebas del SW

1. *Pruebas funcionales - Contrato de Uso:* Se hicieron las pruebas para verificar si el SW cumple con las especificaciones contempladas en el contrato de uso.
2. *Pruebas funcionales -Contrato de Realización.* Se hicieron las pruebas para verificar si el SW cumple con las especificaciones contempladas en el contrato de uso.

7.3.4 Fase 5: Despliegue del SW

Se hizo un diagrama de despliegue de la plataforma donde es desplegado el SW. Ver Figura 7.4

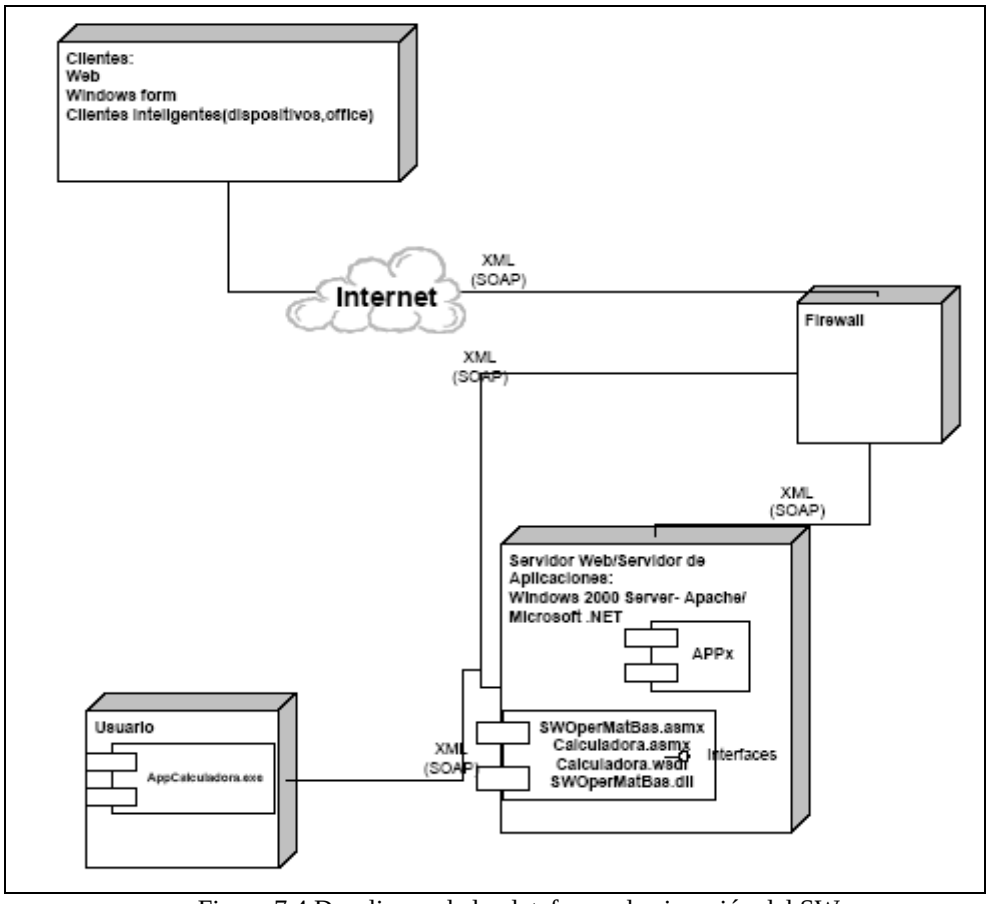

Figura 7.4 Despliegue de la plataforma de ejecución del SW

Se cambió el puerto y URL y se trasladaron los archivos .asmx, wsdl, ,dll perteneciente del ambiente de desarrollo al ambiente de producción, se hizo el despliegue corriente del SW.

7.3.5 Fase 6: Publicación del SW

En este caso, no se registró el servicio en una UDDI, se realizó un archivo HTML donde se muestra el SW SWOperMatBas y su correspondiente archivo WSDL.

CAPÍTULO VIII

CONCLUSIONES Y RECOMENDACIONES

Capítulo que tiene por finalidad mostrar las comparaciones del método DESWeb con otros métodos, ventajas y desventajas del método DESWEb y otros métodos, conclusiones y recomendaciones de la investigación.

El capítulo contempla tres (3) secciones, la sección 8.1 muestra las comparaciones del método DESWeb y otros métodos, la sección 8.2 trata sobre las ventajas y desventajas del método DESWeb y otros métodos y la sección 8.3 se relaciona con las conclusiones y recomendaciones.

8.1 Comparaciones del método DESWeb con otros métodos

Mediante la tabla 8.1 se muestra la comparación del método DESWeb con Los métodos Watch Component, Watch y RUP.

Tabla 8.1 Comparaciones del método DESWeb con otros método (a).

Factor de comparación	**DESWeb**	**Watch Component**	**Watch**	**RUP**
Modelo de proceso desarrollo	1. Especificación del SW. 2. Aprovisionamiento de comp. del SW 3. Ensamblaje SW 4. Pruebas SW 5. Despliegue SW 6. Publicación SW	1. Especificación comp. 2. Diseño comp. 3. Aprovisionamiento comp. 4. Pruebas comp. 5. Certificación comp. 6. Liberación comp.	1. Modelo de Negocios 2. Definición y especificación de requisitos. 3. Diseño arquitectural de aplicación 4. Especificación. De Componentes 5. Aprovisionamiento Comp. 6. Ensamblaje de Componente	1. Inicio 2. Elaboración 3. Construcción 4. Transición

Tabla 8.1 Comparaciones del método DESWeb con otros método (b).

			7. Pruebas 8. Entrega de la aplicación.	
Técnica Empleada	UML	UML	UML	UML
Uso: Didáctica-Empresarial	Ambas	Ambas	Ambas	Ambas
Aplicabilidad	Desarrollo de Servicios Web	Aprovisionamiento de componentes reutilizables	Desarrollo de Aplicaciones Web	Desarrollo de sistemas de software
Alcance	Desde la especificación del SW hasta el despliegue en un repositorio UDDI	Los procesos que cubre van desde la especificación del componente, hasta su colocación dentro de los repositorios.	Desde el modelado del negocio hasta la entrega de la aplicación	Desde el modelado del negocio hasta la entrega de la aplicación
Contexto	Servicio Web	Componentes reutilizables	Aplicaciones Web	Sistemas de software basado en reutilización

8.2 Ventajas y desventajas de los Métodos DESWeb, Watch Component, Watch y Rup

La tabla 8.2 visualiza las ventajas y desventajas de los métodos DESWeb, Watch Component, Watch y RUP.

Tabla 8.2 Ventajas y desventajas de los métodos DESWeb, Watch Component, Watch y Rup (a).

	DESWeb	**Watch Component**	**Watch**	**RUP**
Ventajas	1. Define el proceso de creación de un SW bastante detallada.	1. Define el proceso de creación de un componente reutilizable de manera detallada.	1. Define el proceso de desarrollo de aplicaciones web.	1. Define el proceso de desarrollo de aplicaciones de software basados en

				reutilización.

Tabla 8.2 Ventajas y desventajas de los métodos DESWeb, Watch Component, Watch y Rup (b).

Ventajas	2. Esta orientado a la Arquitectura SOA. 3. Contempla los modelos: producto, proceso, grupo. 4. Se especifican los procesos gerenciales y los de desarrollo. 5. Las fases de los procesos técnicos están bastante detalladas.	2. Modela tanto el producto, el proceso como el grupo de desarrollo. 3. Las fases están definidas hasta el nivel de tareas. 4. Se especifican los procesos gerenciales y los de desarrollo.	2. Contiene los modelos: producto, proceso y grupo 3. Se especifican los procesos gerenciales y los de desarrollo.	2. Modela el proceso y el producto. 3. Definen los diagramas a utilizar.
Desventajas	No se utiliza para desarrollar aplicaciones.	1. No están orientado a la Arquitectura SOA. 2. No se utiliza para desarrollar aplicaciones.	1. No están orientado a la Arquitectura SOA. 2. No cubre la fase de certificación ni de publicación del componente.	No están orientado a la Arquitectura SOA

8.3 Conclusiones y Recomendaciones

SW no es un concepto nuevo, no obstante, es una oportunidad de alcanzar la interoperabilidad de los sistemas actuales. Los SW permiten que se continúe utilizando los

sistemas legados de las organizaciones y que, a su vez, se comuniquen con otras organizaciones, utilizando Internet como canal de comunicación.

La posibilidad de crear SW dentro de las organizaciones (intranet, unidos a otros SW de particulares (extranet), permite lograr la optimización de procesos y operaciones. Los SW representan oportunidades de negocio, tanto para las empresas que desarrollan y publican sus SW, como para las empresas consumidoras de SW.

Aunque los SW ofrecen un modelo de comunicación entre aplicaciones que es independiente de la plataforma, del lenguaje de programación, de la arquitectura H/S, entre otros, dicho modelo implica un gasto en tiempo de aprendizaje de los estándares relacionados. Este aprendizaje es necesario para tener una visión completa de manera de poder ofrecer desarrollos de software de calidad, que permitan insertar mejoras considerables para la organización o integración con organizaciones externas.

Pensamos que un uso sistemático y riguroso de principios, modelos y métodos de Ingeniería de Software para la construcción de SW puede contribuir esencialmente al proceso de desarrollo de pequeñas y medianas organizaciones que quieran mejorar la lógica de negocio. Un proceso de desarrollo de SW puede proceder en varias direcciones y etapas simultáneamente, respetando algún orden parcial en la realización de las tareas. Un método de desarrollo de SW ofrece guías para realizar mejores elecciones, pero no debe prescribir reglas rígidas que sean válidas en todas las circunstancias. Sin embargo, el empleo establecido de un modelo de ciclo de vida puede contribuir favorablemente en el éxito de desarrollo de un proyecto de SW.

En este trabajo de grado, se propuso un nuevo enfoque para representar a los procesos de software y para asistir en la creación y evolución artefactos de SW, denominado Método DESWeb. El método DESWeb abarca desde la fase de especificación hasta el despliegue del SW. Dentro de los logros alcanzados por el método DESWeb, se encuentran:

1. El método DESWeb fue concebido bajo la arquitectura SOA, arquitectura que fue adoptada por los SW.

2. El método DESWeb puede ser el punto de partida para que las pequeñas y medianas organizaciones puedan iniciarse de los SW en un muy poco tiempo y con la oportunidad de ofrecer calidad de servicio.

3. Utilización de la Ingeniería de Métodos como base para la definición del método DESWeb. Esta utilización consta de tres modelos: el modelo de producto, el modelo de procesos técnicos y el modelo de grupo de desarrollo, que se traduce en la interrelación de personas, productos y procesos.

4. Definición de las formas de aprovisionamiento de componentes de SW, el cual puede ser desarrollado, adaptado o contratado.

5. Validación del método DESWeb con un sencillo SW.

En relación a las recomendaciones, tenemos:

1. Aplicación del Método DESWeb en caso de la vida real.
2. Donde se aplique el método DESWeb, se debe crear un repositorio de SW.
3. Donde se aplique el método, se debe adaptarlo con nuevos conceptos que vayan surgiendo en mejoras de la filosofía de SW y en la Ingeniería del Software.

4. Utilizar herramientas CASE en las diferentes fases del método DESWeb, con la finalidad de aumentar la productividad en el desarrollo de los SW, de manera de reducir costos en términos de tiempo y dinero.

BIBLIOGRAFÍA

Bussler, C. The ACM Digital Library. 2002. Software as a service: ASP and ASP aggregation. <http://portal.acm.org/citation.cfm?id=564801&coll=GUIDE&dl=ACM&CFID=15151515&CFTOKEN=6184618> (junio, 2006).

Chartier, R. Application Architecture: An N-Tier Approach - Part 1. 2001.< http://www.15seconds.com/issue/011023.htm> (Julio, 2005).

Cheesman, J. & Daniels, J. (2001). *UML Component. A simple process for specifying Component Based Software*. EEUU: Addison Wesley. (2001).

Cubillos, J., Burbano, J., Corrales, J., Ordóñez, J. 2004. Composición Semántica de Servicios Web. < http://www.cintel.org.co/media/temacentral_3_14.pdf>.(septiembre, 2005).

Erl, T. (2004). *Service- Oriented Architecture*. EEUU: Prentice Hall PTR. (2004). Lenguaje de descripción de servicios Web (WSDL) 1.0. 2003 < http://www.microsoft.com/spain/msdn/articulos/archivo/090201/voices/wsdl.asp> (Abril 2004).

Hamar, Vanessa. «*Aspectos metodológicos del desarrollo y reutilización de componentes de software*». Mérida - Venezuela: Universidad de Los Andes. Facultad de Ingeniería Postgrado en Computación. 2003. Tesis Maestría.

Han, Jun. 1999. citado por Hamar, Vanessa. «*Aspectos metodológicos del desarrollo y reutilización de componentes de software*». Mérida - Venezuela: Universidad de Los Andes. Facultad de Ingeniería Postgrado en Computación. 2003. Tesis Maestría.

Hilliard, R. The ACM Digital Library. 2001. IEEE Std 1471 and Beyond*. <http://www.enterprise-architecture.info/Images/Documents/IEEE%201471-%20Beyond.pdf> (julio, 2006).

Kruchten P. *The 4+1 View Model of Architecture. IEEE Computer Society Digital Library*November 1995 (Vol. 12, No. 6) pp. 42-50. http://doi.ieeecomputersociety.org/10.1109/52.469759. (Marzo, 2006).

Lizárraga, C. Universal Description, Discovery, and Integration. 2002 < http://www.fisica.uson.mx/carlos/WebServices/WS_UDDI.htm > (Abril 2004).

Montilva, J. And Barrios, J. A Component-Based Method for Developing Web Applications. Revista Colombiana de Computación (Colombian Journal of Computation). Vol. 4, No.1, July, 2003, pp. 21- 34.

Nandigam, J., Gudivada, V. Kalavala, M. ACM Digital Library. 2005. SEMANTIC WEB AND WEB SERVICES. < http://delivery.acm.org/10.1145/1090000/1088801/p50-nandigam.pdf?key1=1088801&key2=7445092511&coll=GUIDE&dl=ACM&CFID=15151515&CFTOKEN=6184618 > (junio, 2005).

Odell, J.J., 1996. citado por Jonas A. Montilva C. and Judith Barrios A en el artículo "a component-Based METHOD FOR Developing web applications". A Primer to

Method Engineering. *INFOSYS: The electronic newsletter for information systems.* 3 (19).

Peltz, C, 2003. Web Services Orchestration. < http://devresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf > (marzo, 2004).

PC-NEWS. ASP (Application Service Provider) .2002. <http://www.pc-news.com/detalle.asp?sid=&id=44&Ida=709 > (mayo,2006).

Ramírez M. Arachnida. 2005. ¿Qué son Servicios-WEB?. <http://www.arachnida.com/view/index.asp?ms=78&pageMs=4537> (marzo 2005).

Sanchez. D. Schenone. 2004. < www.codejava.org/files/checkfile.php?file=/ Codejava_31_Herramientas.doc>. (enero, 2005).

Short, S. (2002). *Creación de Servicios WEB XML para la Plataforma Microsoft.NET* . Madrid: McGrawHill . (2002).

Tarsis. Application Service Provider (ASP): Un modelo de servicio. 2006. < http://www.tarsis.net/?page=asp> (mayo,2006).

Topxml, Aprenda XML. < http://www.topxml.com/xml/learn/learnxml.sp.asp> (mayo 2005).

W3C. Simple Object Access Protocol (SOAP) 1.1. 2000. <http://www.w3.org/TR/SOAP/> (Mayo 2005).

W3C. Web Services Architecture. 2004. <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>(marzo de 2004).

Weerawarana, S., Curbera, F, Leymann F, Storey, T, Fergusan. D. (2005). Web *Services Platform Architecture*. EEUU: Prentice Hall PTR. Service Composition. (p.314).

Wikipedia. Ingeniería de software. <http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software> (Junio,2005).

Wikipedia. Proveedor de servicios de aplicación. 2006. < http://es.wikipedia.org/wiki/Application_Service_Provider> (mayo,2006).

Wikipedia . Programación Orientada a Aspectos. <http://es.wikipedia.org/wiki/Programaci%C3%B3n_Orientada_a_Aspectos> (junio,2006).

Zimmermann y Muller. SOA Project Planning Aspects. 2004. www.ibmpressbooks.com/articles/article.asp?p=422305&seqNum=5&r1=1 (julio,2006).